Technische Universität Ilmenau
Institut für Mathematik

Preprint No. M 00/11

MATLAB - Teil IV; Approximation, Numerische Integration

Neundorf, Werner

2000

Impressum:
Hrsg.: Leiter des Instituts für Mathematik
Weimarer Straße 25
98693 Ilmenau
Tel.: +49 3677 69 3621
Fax: +49 3677 69 3270
http://www.tu-ilmenau.de/ifm/

ISSN xxxx-xxxx

ilmedia

Zusammenfassung

This is a tutorial on teaching and programming in MATLAB.
It is based on scripts and exercises in the course of numerical mathematics for students of the faculties Electrical Engineering and Information Technology and Computer Science and Automation after first term.
The part I contains basic aspects and elements of numerical linear algebra, especially methods for systems of linear equations. The second part gives some aspects about storage and import/export of data files, furthermore MATLAB-based algorithms and programming tools for special systems of linear equations, for eigenvalue problems and singular value decomposition, graphic aspects, nonlinear equations and systems of equations. The third part concerns systems of complex linear equations, polynomial and spline interpolations.
In this part IV we consider the sections of approximation and numerical integration.

Vorwort

MATLAB is an interactive, matrix-based system for scientific and engineering calculations. You can solve complex numerical problems without actually writing a program. The name MATLAB is an abbreviation for MATrix LABoratory.

A few words to those who are familiar with other programming languages.

- MATLAB is a user-friendly high-level programming language and important technical computing environment. It also includes a number of functional programming constructs, modeling, simulation and prototyping, application development and design.

- MATLAB has a rich environment of powerful toolboxes and data visualization. It offers programming structures like control flows, selections, decisions and m-files functions.

- Students can affordably use this powerful numeric computation, data analysis and visualization software in their undergraduate and graduate studies. The student edition encapsulates a wide range of disciplines.

- MATLAB is not strongly typed like C and Pascal. No declarations are required. It is more like Basic and Lisp in this respect. You can dynamically link C or FORTRAN subroutines. Some type checking is done at run time.

- MATLAB suitable for running numerically intensive programs with double-precision numerical calculations. On the other site there are, based on Maple V, many symbolic tools and symbolic functions to combine, simplify, differentiate, integrate, and solve algebraic and differential equations. The symbolic toolbox and the subroutines concept of MATLAB seems to be not so efficiently as is described in Maple.

- MATLAB is available for a number of environments: Sun/Apollo/VAXstation/HP workstations, VAX, MicroVAX, Gould, PC, Apple Macintosh, and several parallel machines.

The aim is to show how you can write simple instructions, commands or programs in MATLAB for doing numerical calculations, linear algebra, and programs for simplifying or transforming expressions, equations, mathematical formulas or arrays.

It is assumed that the reader is familiar with using MATLAB interactively. For beginners we propose the introductional tutorial, the so called Primer. The purpose of this Primer based on the version 3.5 is to help you begin to use MATLAB. They can best be used hands-on. You are encouraged to work at the computer as you read the Primer and freely experiment with examples.

This document is based on **MATLAB version 4.2c1**.
MATLAB development continues. New versions come out every one or two years which contain not only changes to the mathematical capabilities of MATLAB, but also changes to the programming language and user interface. The MATLAB 5.2 and 5.3 highlights you can find on the web site

`http://www.mathworks.com/products/matlab/highlights.shtml` .

We are pleased and somewhat surprised to see how quickly this movement is already happening. For this reason, I have applied constructs in the language that will be probable in the language in future versions of MATLAB.

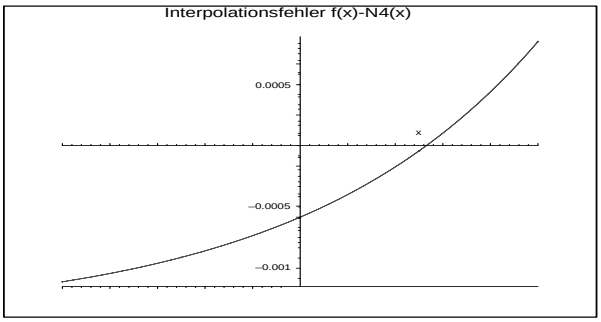
You should liberally use the on-line help facility for more detailed information. After entering MATLAB the command `help` will display a list of functions for which on-line help is available. The command `help` *functionname* will give information about a specific function. You can preview some of the features of MATLAB by entering the command `demo`. Even better, access help from the menu.
In the bibliography there are given some useful sources of information and supplemental workbooks for MATLAB.

Inhaltsverzeichnis

1 Approximation

1.1 Interpolation

Die Interpolation ist eine spezielle Form der Approximation.
Für die Funktion $f(x) = e^x$ und der Referenz mit den $n+1 = 5$ Stützstellen $x_j : \pm 1, \pm\frac{1}{2}, 0$, und Stützwerten $y_j = f(x_j)$ erhalten wir auf dem Intervall [-1,1] das Interpolationspolynom $N_4(x) = 0.043435x^4 + 0.177348x^3 + 0.499645x^2 + 0.997853x + 1$ und seinen Fehler.

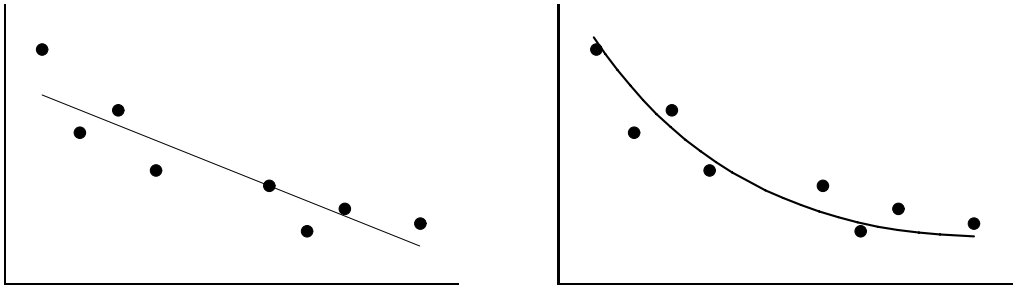


1.2 Diskrete Approximation im Mittel

Problemstellung

- Zwischen zwei Größen x und y wird ein funktionaler Zusammenhang $y = f(x)$ vermutet, der jedoch häufig nur durch Messungen an den *Meßpunkten* $x_j, j = 0(1)N$, bekannt ist.

- Eine graphische Darstellung der *Meßwerte* y_j über den Meßpunkten gibt mitunter Aufschluß über den *Typ* der zugrundeliegenden Funktion. In der Abbildung wird nicht eine lineare Funktion wie in der linken Abbildung, sondern eine quadratische Funktion vermutet, d.h. ein Ansatz der Form $\varphi(x) = a_0 + a_1 x + a_2 x^2$ ist sinnvoll.



- Bei der *numerischen Approximation* wird eine konkrete Funktion $\varphi(x)$ aus einer vorgegebenen Funktionenklasse so durch die "Punktwolke" der Meßwerte (x_j, y_j) hindurchgelegt, so daß ihr *Abstand* zu diesen Meßpunkten insgesamt minimal wird. Der Abstandsbegriff kann sich sowohl auf das Lot als auch die Entfernung in vertikaler oder horizontaler Richtung beziehen.

- Die Punktwolke der Meßwerte kann auch allgemeiner beschrieben sein (mehrdimensionaler Fall).

1.2.1 Methode der kleinsten Quadrate

Aufgabenstellung

- Abgeschlossene, beschränkte Grundmenge $D \subset \mathbb{R}^r$, $r \geq 1$, und eine unbekannte reelle Funktion $f : D \to \mathbb{R}$.

- Stützstellen x_j und Stützwerte y_j

$$R = \{x_j : \quad x_j \quad = \quad (x_{j1}, x_{j2}, ..., x_{jr}) \in D, \\ x_j \text{ nicht unbedingt paarweise verschieden, } j = 0(1)N\}$$

  eine *Referenz* mit $N+1$ *Stützstellen (Meßpunkten)* und den $N+1$ zugehörigen *Stützwerten (Meßwerten)* $y_j = f(x_j)$, $j = 0(1)N$.

- Ansatzfunktion $\varphi(x)$ der Form

$$\varphi(x) = a_0\varphi_0(x) + a_1\varphi_1(x) + ... + a_n\varphi_n(x), \quad a_i \in \mathbb{R},$$

  mit vorgegebenen $n+1$ *Basisfunktionen* $\varphi_i(x)$, $i = 0(1)n$.

- Approximationsforderung (Approximationsbedingung)

$$F = F(a_0, a_1, ..., a_n) = \frac{1}{2}\sum_{j=0}^{N} (y_j - \varphi(x_j))^2 \quad \longrightarrow \quad \min.$$

- Approximationsaufgabe
  Gesucht sind Koeffizienten $a_0, a_1, ..., a_n$ einer *approximierenden Funktion* $\varphi(x)$, so daß der Wert F minimal ist.

Typen von Ansatzfunktionen

- Allgemeine lineare Approximation

$$\varphi(x) = a_0\varphi_0(x) + a_1\varphi_1(x) + ... + a_n\varphi_n(x)$$

  mit *Basisfunktionen* $\varphi_i(x)$.

- Approximation durch Polynome in $\mathbb{R}^1$

$$\varphi(x) = a_0 + a_1x + a_2x^2 + ... + a_nx^n, \quad a_i \in \mathbb{R},$$

  mit Basisfunktionen $\varphi_i(x) = x^i$.

  Spezialfälle

$$\begin{aligned} \varphi(x) &= a_0 + a_1x && \text{(linearer Ausgleich)} \\ \varphi(x) &= a_0 + a_1x + a_2x^2 && \text{(quadratischer Ausgleich)} \end{aligned}$$

- Approximation durch trigonometrische Polynome (Beispiel)

$$\varphi(x) = a_0 + a_1 \cos x + a_2 \sin x.$$

- Lineare Approximation in $\mathbb{R}^n$

$$\varphi(x_1, x_2, ..., x_n) = a_0 + a_1 x_1 + a_2 x_2 + ... + a_n x_n$$

mit $x = (x_1, x_2, ..., x_n)$.

3 Fragestellungen

1. Existiert zu jeder gegebenen Referenz R mit $N+1$ Punkten eine approximierende Funktion $\varphi(x)$ aus der gegebenen Funktionenklasse ?
2. Ist $\varphi(x)$ eindeutig bestimmt?
3. Wie kann $\varphi(x)$ effektiv konstruiert werden?

Herleitung der Methode der kleinsten Quadrate von C.F.Gauß

Gesucht ist ein Koeffizientensatz $a_0, a_1, ..., a_n$, für den die Summe der Abweichungsquadrate, die *Ausgleichsfunktion*

$$F(a_0, a_1, ..., a_n) = \frac{1}{2} \sum_{j=0}^{N} \left[y_j - \sum_{i=0}^{n} a_i \varphi_i(x_j) \right]^2$$

minimal wird.
Notwendige Bedingungen für ein (lokales) Minimum sind

$$\frac{\partial F}{\partial a_k} = \sum_{j=0}^{N} \left[\sum_{i=0}^{n} a_i \varphi_i(x_j) - y_j \right] \varphi_k(x_j) = 0, \;\; k = 0(1)n.$$

Auflösung der Klammern und Umstellen liefert

$$\sum_{i=0}^{n} \left[\sum_{j=0}^{N} \varphi_k(x_j) \varphi_i(x_j) \right] a_i \quad = \quad \sum_{j=0}^{N} \varphi_k(x_j) y_j, \quad k = 0(1)n.$$

Durch Einführung der Gaußschen Klammern (Skalarprodukte, Gewicht $\omega = 1$)

$$\begin{aligned} (\varphi_k, \varphi_i) &= \sum_{j=0}^{N} \varphi_k(x_j) \varphi_i(x_j) \\ (\varphi_k, y) &= \sum_{j=0}^{N} \varphi_k(x_j) y_j, \quad \varphi_k, y \in l_{2,1} \end{aligned}$$

lassen sich die $n+1$ Bedingungen für die $n+1$ Unbekannten a_i in der Form der *Normalgleichungen* notieren.

$$\begin{aligned}
(\varphi_0,\varphi_0)a_0 + (\varphi_0,\varphi_1)a_1 + \cdots + (\varphi_0,\varphi_n)a_n &= (\varphi_0,y)\\
(\varphi_1,\varphi_0)a_0 + (\varphi_1,\varphi_1)a_1 + \cdots + (\varphi_1,\varphi_n)a_n &= (\varphi_1,y)\\
&\cdots\cdots\cdots\cdots\\
(\varphi_n,\varphi_0)a_0 + (\varphi_n,\varphi_1)a_1 + \cdots + (\varphi_n,\varphi_n)a_n &= (\varphi_n,y)
\end{aligned}$$

Die Determinante der Koeffizientenmatrix

$$det(G) = det(G(\varphi_0,...,\varphi_n)) = det((\varphi_i,\varphi_j)_{i,j=0}^n)$$

heißt auch GRAMsche Determinante.

Definition **Verallgemeinerte HAARsche Bedingung**

Die $n+1$ Basisfunktionen $\varphi_i(x)$, $i = 0(1)n$, erfüllen die *verallgemeinerte HAARsche Bedingung*, falls für jede Referenz $R \subset D$ die HAARsche oder Vandermondesche Matrix

$$\Phi = \begin{pmatrix}
\varphi_0(x_0) & \varphi_1(x_0) & \varphi_2(x_0) & \cdots & \varphi_n(x_0)\\
\varphi_0(x_1) & \varphi_1(x_1) & \varphi_2(x_1) & \cdots & \varphi_n(x_1)\\
\varphi_0(x_2) & \varphi_1(x_2) & \varphi_2(x_2) & \cdots & \varphi_n(x_2)\\
\cdots & \cdots & \cdots & \cdots & \cdots\\
\varphi_0(x_N) & \varphi_1(x_N) & \varphi_2(x_N) & \cdots & \varphi_n(x_N)
\end{pmatrix}$$

den maximalen Rang $r = \min(n,N)+1$ besitzt.

Das Normalgleichungssystem notieren wir damit in der Kurzform

$$Ga \;=\; \Phi^T\Phi a \;=\; \Phi^T y, \quad G \text{ Koeffizientenmatrix.}$$

Existenz und Eindeutigkeit

Falls $N \geq n$ und die verallgemeinerte HAARsche Bedingung erfüllt ist, liefern die Normalgleichungen das absolute Minimum der Ausgleichsfunktion $F(a_0,a_1,...,a_n)$. Die Approximationsaufgabe ist für beliebige Referenzen $R \subset D$ *stets eindeutig lösbar.*

Bemerkungen

(1) Falls $N \geq n$ ist müssen zur Sicherung von $rang(\Phi) = n+1$ unter den $N+1$ Stützstellen x_j mindestens $n+1$ verschiedene sein.
Der Fall $N < n$ führt auf ein unterbestimmtes System (zu wenig Bedingungen) und wird i.a. wegen der Mehrdeutigkeit der Lösung nicht weiter betrachtet.

(2) Sei $N = n$. Das System von Basisfunktionen $\{\varphi_i(x)\}$, das mit einer beliebigen Referenz mit paarweise verschiedenen Stützstellen eine stets reguläre Matrix Φ bildet, heißt auch *Haarsches*, Tschebyscheff-, T- oder unisolventes System. Die Aufgabe ist dann wie ein Interpolationsproblem eindeutig lösbar.

(3) Sei $N \geq n$. Analoge Aussage zu (2) mit Referenz, die mindestens $n+1$ verschiedene Stützstellen enthalten muß. Die Regularität von Φ wird ersetzt durch $rang(\Phi) = n+1$ oder $det(G) \neq 0$.

Haarsche Funktionensysteme

- $\{1, x, ..., x^n\}$ über $[a,b] \subset \mathbb{R}$
- $\{1, \cos(kx), \sin(kx)\},\ k = 1, 2, ..., K$, über $[0, 2\pi]$

Berechnung des Approximationsfehlers

Unter Berücksichtigung der Erfüllung des Normalgleichungssystems durch die Koeffizienten $a_i,\ i = 0, 1, ..., n$, ergibt sich die Fehlerformel

$$\begin{aligned} F_{min} &= \frac{1}{2}\sum_{j=0}^{N}(y_j - \varphi(x_j))^2, \qquad \varphi(x_j) = \sum_{i=0}^{n} a_i\varphi_i(x_j) \\ &= \frac{1}{2}\Big[\sum_{j=0}^{N} y_j^2 - \sum_{i=0}^{n} a_i(\varphi_i, y)\Big] \ \geq\ 0. \end{aligned}$$

Beispiel

Sei $N = n$ und $p(x)$ eine beliebige Funktion, die im betrachteten Stützstellenbereich keine Nullstelle besitzt. Dann bilden die Funktionen

$$\frac{1}{p(x)},\ \frac{x}{p(x)},\ \frac{x^2}{p(x)},\ ...,\frac{x^n}{p(x)}$$

ein Haarsches System. Wir testen dies auf [1,4] für $n = 2$ und $p(x) = 1 + e^{-x}$.

```
disp('Haarsche Bedingung')
% Definition der Matrix
n = 2
ma = zeros(n+1);
xx = [ 1 2 4 ]
p = '1+exp(-x)';      % symbolische Definition
px = numeric(subs(p,xx(1),'x'))    % Kontrolle

% Numerische Rechnung
for i=1:n+1
  xh = xx(i);
  px = numeric(subs(p,xh,'x'));
  ah = 1/px;
  for j=1:n+1
    ma(i,j) = ah;
    ah = ah*xh;
  end;
end;
```

```
ma
disp('Haarsche Determinante <> 0 ?')
HB = det(ma)
```

Ergebnisse

```
Haarsche Bedingung
n =
     2

xx =
     1     2     4

px =
    1.3679

ma =
    0.7311    0.7311    0.7311
    0.8808    1.7616    3.5232
    0.9820    3.9281   15.7122

Haarsche Determinante <> 0 ?

HB =
    3.7940
```

Bei der zweiten Variante der Berechnung der Determinante der Haarschen Matrix mittels symbolischer Umformungen verwenden wir zunächst symbolische Kommandos wie

- `sym` Create, access, or modify a symbolic matrix,
- `determ` Symbolic matrix determinant,
- `factor` Symbolic factorization,

um dann mit dem Substitutionsbefehl `subs` die endgültige numerische Auswertung vorzunehmen. Man erkennt nach der Faktorisierung die Vandermondesche Determinante

$$\frac{(x(3)-x(1))*(x(2)-x(1))*(x(3)-x(2))}{p(x(1))*p(x(2))*p(x(3))}.$$

```
disp('Test')
p1 = subs(p,xx(1),'x')
p2 = subs(p,xx(2),'x')
p3 = subs(p,xx(3),'x')

T1 = sym('[ a, b ; c, d ]')
T2 = sym(3,3,'(i+j)/(i-j+s)')
```

```
disp('Symbolische Rechnung')
% R1 in einer Zeile notieren
R1 = sym('[ 1/p(x(1)), x(1)/p(x(1)), x(1)^2/p(x(1));
            1/p(x(2)), x(2)/p(x(2)), x(2)^2/p(x(2));
            1/p(x(3)), x(3)/p(x(3)), x(3)^2/p(x(3)) ]')
D = determ(R1)
DF = factor(D)

disp('Haarsche Determinante <> 0 ?')
disp(' ')
disp('p(x(i)) einsetzen')
DF1 = subs(DF, subs(p,xx(1),'x'),'p(x(1))');
DF2 = subs(DF1,subs(p,xx(2),'x'),'p(x(2))');
DF3 = subs(DF2,subs(p,xx(3),'x'),'p(x(3))')
disp(' ')
disp('x(i) einsetzen')
EF1 = subs(DF3,xx(1),'x(1)');
EF2 = subs(EF1,xx(2),'x(2)');
EF3 = subs(EF2,xx(3),'x(3)')

disp(' ')
disp('Numerische Auswertung (Ergebnis wie oben mit det)')
HB1 = numeric(EF3)
```

Ergebnisse

```
Test

p1 =
1+exp(-1)

p2 =
1+exp(-2)

p3 =
1+exp(-4)

T1 =
[ a, b ]
[ c, d ]

T2 =
[     2/s, 3/(-1+s), 4/(-2+s) ]
[ 3/(1+s),      4/s, 5/(-1+s) ]
[ 4/(2+s),  5/(1+s),      6/s ]
```

```
Symbolische Rechnung

R1 =
[ 1/p(x(1)), x(1)/p(x(1)), x(1)^2/p(x(1)) ]
[ 1/p(x(2)), x(2)/p(x(2)), x(2)^2/p(x(2)) ]
[ 1/p(x(3)), x(3)/p(x(3)), x(3)^2/p(x(3)) ]


D =
-(-x(2)*x(3)^2+x(2)^2*x(3)+x(1)*x(3)^2-x(1)^2*x(3)-x(1)*x(2)^2
  +x(1)^2*x(2))/p(x(1))/p(x(2))/p(x(3))

DF =
-(x(3)-x(1))*(x(2)-x(1))*(-x(3)+x(2))/p(x(1))/p(x(2))/p(x(3))

Haarsche Determinante <> 0 ?

p(x(i)) einsetzen

DF3 =
-(x(3)-x(1))*(x(2)-x(1))*(-x(3)+x(2))/(1+exp(-1))/(1+exp(-2))/(1+exp(-4))

x(i) einsetzen

EF3 =
6/(1+exp(-1))/(1+exp(-2))/(1+exp(-4))

Numerische Auswertung (Ergebnis wie oben mit det)

HB1 =
    3.7940
```

1.2.2 Ausgleich durch Polynome in $\mathbb{R}^1$

Ansatzfunktion

$$\varphi(x) = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n, \quad a_i \in \mathbb{R},$$

mit Basisfunktionen (Monome) $\varphi_i(x) = x^i,\ i = 0(1)n.$

Spezialfälle

$$\begin{aligned}
\varphi(x) &= a_0 + a_1 x && \text{(linearer Ausgleich)}\\
\varphi(x) &= a_0 + a_1 x + a_2 x^2 && \text{(quadratischer Ausgleich)}\\
\varphi(x) &= a_0 + a_1 x + a_2 x^2 + a_3 x^3 && \text{(kubischer Ausgleich)}
\end{aligned}$$

Normalgleichungen

Mit den Gaußschen Klammern (Skalarprodukten)

$$\begin{aligned}
(\varphi_k, \varphi_i) &= \sum_{j=0}^{N} x_j^k x_j^i = \sum_{j=0}^{N} x_j^{k+i},\\
(\varphi_k, y) &= \sum_{j=0}^{N} x_j^k y_j, \quad i,k = 0(1)n,
\end{aligned}$$

lassen sich die $n+1$ Normalgleichungen für die $n+1$ Unbekannten a_i in der folgenden Form notieren.

$$\begin{pmatrix}
N+1 & \sum x_j & \sum x_j^2 & \cdots & \sum x_j^n\\
\sum x_j & \sum x_j^2 & \sum x_j^3 & \cdots & \sum x_j^{n+1}\\
\sum x_j^2 & \sum x_j^3 & \sum x_j^4 & \cdots & \sum x_j^{n+2}\\
\cdots & \cdots & \cdots & \cdots & \cdots\\
\sum x_j^n & \sum x_j^{n+1} & \sum x_j^{n+2} & \cdots & \sum x_j^{2n}
\end{pmatrix}
\begin{pmatrix} a_0\\ a_1\\ a_2\\ \cdots\\ a_n \end{pmatrix}
=
\begin{pmatrix} \sum y_j\\ \sum x_j y_j\\ \sum x_j^2 y_j\\ \cdots\\ \sum x_j^n y_j \end{pmatrix}$$

Existenz und Eindeutigkeit

Falls $N \geq n$ ist und mindestens $n+1$ Stützstellen x_j verschieden sind, so liefern die Normalgleichungen das absolute Minimum der Ausgleichsfunktion $F(a_0, a_1, ..., a_n)$.
Die Approximationsaufgabe ist für beliebige Intervalle $I = [a, b]$ und beliebige Referenzen $R \subset I$ mit mindestens $n+1$ verschiedene Stützstellen *stets eindeutig lösbar.*

Lösung des Normalgleichungssystems

Zunächst beachte man, daß die Kondition der Koeffizientenmatrix G durchaus schlecht werden kann und $det(G) = det(\Phi^T \Phi) \approx 0$ für nahe beieinander liegende Stützstellen ist.

Im allgemeinen Fall sind zum Aufstellen des LGS $(n+1)(n+2)$ Skalarprodukte auszuwerten. Dazu dient das FALKsche Schema.

					1	x_N	x_N^2	$\cdots$	x_N^n	y_N
					1	x_{N-1}	x_{N-1}^2	$\cdots$	x_{N-1}^n	y_{N-1}
					$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
					1	x_1	x_1^2	$\cdots$	x_1^n	y_1
					1	x_0	x_0^2	$\cdots$	x_0^n	y_0
1	1	...	1	1	$N+1$	$\sum x_j$	$\sum x_j^2$	$\cdots$	$\sum x_j^n$	$\sum y_j$
x_N	x_{N-1}	...	x_1	x_0	$\sum x_j$	$\sum x_j^2$	$\sum x_j^3$	$\cdots$	$\sum x_j^{n+1}$	$\sum x_j y_j$
x_N^2	x_{N-1}^2	...	x_1^2	x_0^2	$\sum x_j^2$	$\sum x_j^3$	$\sum x_j^4$	$\cdots$	$\sum x_j^{n+2}$	$\sum x_j^2 y_j$
.....					.....					.....
x_N^n	x_{N-1}^n	...	x_1^n	x_0^n	$\sum x_j^n$	$\sum x_j^{n+1}$	$\sum x_j^{n+2}$	$\cdots$	$\sum x_j^{2n}$	$\sum x_j^n y_j$

Wegen der Symmetrie und gleicher Summen kann man den Aufwand reduzieren, so daß nur $2n + (n+1) = 3n + 1$ Skalarprodukte bzw. Summen zu bestimmen sind.
Folgendes Rechenschema ist günstiger.

j	x	y	xy	x^2	x^2y	x^3	x^3y	x^4	$\cdots$	x^{2n}
0	x_0	y_0	x_0y_0	x_0^2	$x_0^2y_0$	x_0^3	$x_0^3y_0$	x_0^4	$\cdots$	x_0^{2n}
1	x_1	y_1	x_1y_1	x_1^2	$x_1^2y_1$	x_1^3	$x_1^3y_1$	x_1^4	$\cdots$	x_1^{2n}
2	x_2	y_2	x_2y_2	x_2^2	$x_2^2y_2$	x_2^3	$x_2^3y_2$	x_2^4	$\cdots$	x_2^{2n}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
N	x_N	y_N	x_Ny_N	x_N^2	$x_N^2y_N$	x_N^3	$x_N^3y_N$	x_N^4	$\cdots$	x_N^{2n}
$N+1$	$\sum x_j$	$\sum y_j$	$\sum x_jy_j$	$\sum x_j^2$	$\sum x_j^2y_j$	$\sum x_j^3$	$\sum x_j^3y_j$	$\sum x_j^4$	$\cdots$	$\sum x_j^{2n}$

Linearer Ausgleich

Lineare Ansatzfunktion $\varphi(x) = a_0 + a_1 x$

Normalgleichungssystem

$$\begin{aligned} a_0(N+1) + a_1 \sum_{j=0}^{N} x_j &= \sum_{j=0}^{N} y_j \\ a_0 \sum_{j=0}^{N} x_j + a_1 \sum_{j=0}^{N} x_j^2 &= \sum_{j=0}^{N} x_j y_j \end{aligned}$$

mit der Koeffizientendeterminante

$$D = (N+1)\sum_{j=0}^{N} x_j^2 - \left(\sum_{j=0}^{N} x_j\right)^2 \neq 0$$

Lösung mit CRAMERscher Regel

$$a_0 = \left(\sum_{j=0}^{N} y_j \sum_{j=0}^{N} x_j^2 - \sum_{j=0}^{N} x_j \sum_{j=0}^{N} x_j y_j \right) / D$$

$$a_1 = \left((N+1) \sum_{j=0}^{N} x_j y_j - \sum_{j=0}^{N} x_j \sum_{j=0}^{N} y_j \right) / D$$

Quadratischer Ausgleich

Quadratische Ansatzfunktion $\varphi(x) = a_0 + a_1 x + a_2 x^2$

Normalgleichungssystem

$$\begin{aligned} a_0 (N+1) + a_1 \sum_{j=0}^{N} x_j + a_2 \sum_{j=0}^{N} x_j^2 &= \sum_{j=0}^{N} y_j \\ a_0 \sum_{j=0}^{N} x_j + a_1 \sum_{j=0}^{N} x_j^2 + a_2 \sum_{j=0}^{N} x_j^3 &= \sum_{j=0}^{N} x_j y_j \\ a_0 \sum_{j=0}^{N} x_j^2 + a_1 \sum_{j=0}^{N} x_j^3 + a_2 \sum_{j=0}^{N} x_j^4 &= \sum_{j=0}^{N} x_j^2 y_j \end{aligned}$$

Lösung mit CRAMERscher Regel sowie SARRUSscher Regel zur Determinantenberechnung.

Die Kondition des entstehenden Normalgleichungssystems wird mit wachsender Dimension n immer schlechter. Um z.B. große Elemente der Koeffizientenmatrix zu vermeiden, empfiehlt sich eine Koordinatentransformation der Stützstellen x_j.
Damit erhalten wir ein skaliertes System mit besseren Eigenschaften.

1.2.3 Die MATLAB Funktion `polyfit`

MATLAB stellt für die Methode der kleinsten Quadrate mit Polynomen das m-File `polyfit` zur Verfügung.

```
p = polyfit(x,y,n)
```

Die Funktion liefert in Form eines Zeilenvektors die Koeffizienten $p(1 \,..\, n+1)$ des Ausgleichspolynoms $\varphi(x) = \sum_{i=0}^{n} a_i x^i$ vom Grade n zu den Daten x, y, also $p(i) = a_{n+1-i}$, $i = 1, 2, ..., n+1$.
Aus diesen Koeffizienten kann man in einem ausgewählten Intervall den Verlauf des Ausgleichspolynoms mittels `polyval` erzeugen. Numerische und graphische Vergleiche von $\varphi(x)$ mit $y = f(x)$ (falls y durch eine Funktion generiert wurde) sind möglich.

In `polyfit` wird nicht das Normalgleichungssystem aufgestellt, sondern das System mit der rechteckigen Haarschen Matrix Φ.

$$\sum_{i=n(-1)}^{0} x_j^i a_i \sim y_j, \quad j = 0, 1, ..., N.$$

In der Links-Division $x = A\backslash b$ wird die quadratische Matrix mittels Gaußelimination faktorisiert und damit dann das LGS $Ax = b$ gelöst.
Für eine rechteckige Matrix A liegt die Householder-Orthogonalisierung (QR-Verfahren) zugrunde, und man sucht die Lösung im Sinne der Methode der kleinsten Quadrate (least squares sense). Die Funktion in der MATLAB Toolbox ist folgende.

```
function [p,S] = polyfit(x,y,n)

%POLYFIT Polynomial curve fitting.
% p = POLYFIT(x,y,n) finds the coefficients of a polynomial p(x) of
% degree n that fits the data, p(x(i)) ~= y(i), in a least-squares sense.
%
% [p,S] = POLYFIT(x,y,n) returns the polynomial coefficients p and a
% matrix S for use with POLYVAL to produce error estimates on predictions.
% If the errors in the data, y, are independent normal with constant
% variance, POLYVAL will produce error bounds which contain at least 50%
% of the predictions.
%
% See also POLY, POLYVAL, ROOTS.

% J.N. Little 4-21-85, 8-23-86; CBM, 12-27-91 BAJ, 5-7-93.
% Copyright (c) 1984-94 by The MathWorks, Inc.

% The regression problem is formulated in matrix format as:
%
%    y = V*p    or
%
%          3  2
%    y = [x  x  x  1] [p3
%                      p2
%                      p1
%                      p0]
%
% where the vector p contains the coefficients to be found.  For a
% 7th order polynomial, matrix V would be:
%
% V = [x.^7 x.^6 x.^5 x.^4 x.^3 x.^2 x ones(size(x))];

if any(size(x) ~= size(y))
    error('X and Y vectors must be the same size.')
end

x = x(:);
y = y(:);
```

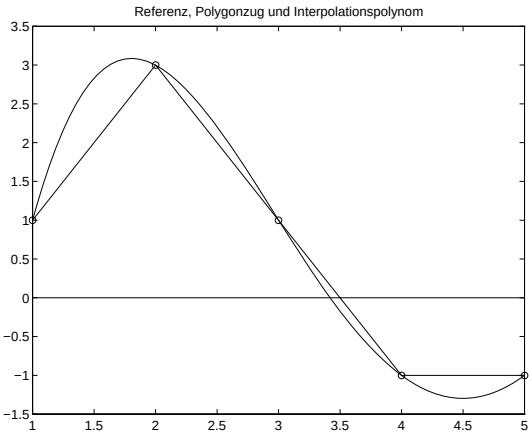
```
% Construct Vandermonde matrix.
V(:,n+1) = ones(length(x),1);
for j = n:-1:1
    V(:,j) = x.*V(:,j+1);
end

% Solve least squares problem.
[Q,R] = eval('qr(V,0)','qr(V)');

% The current PC version does not have the two-argument form of qr
[rows, cols] = size(R);
if rows ~= cols
   R = R(1:cols,:);
   Q = Q(:,1:cols);
end

p = R\(Q'*y);    % Same as p = V\y;
r = y - V*p;
p = p';          % Polynomial coefficients are row vectors by convention.

% S is a structure containing three elements: the Cholesky factor of the
% Vandermonde matrix, the degrees of freedom and the norm of the residuals.

df = length(y) - (n+1);
S = [R; [df zeros(1,n)]; [norm(r) zeros(1,n)]];
```

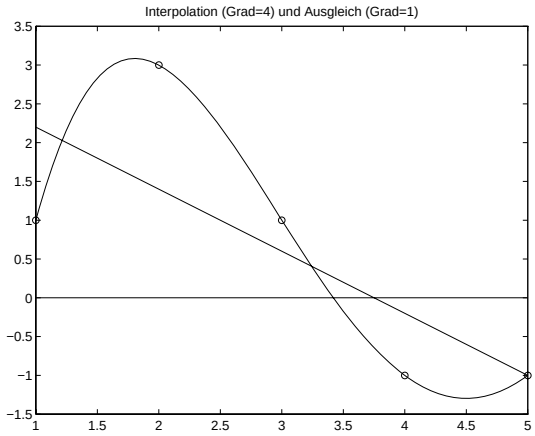
Beispiel 1

Gegeben sei eine Referenz mit paarweise verschiedenen Stützstellen.
Berechne dazu das Interpolationspolynom und teste die verschiedenen Möglichkeiten zur Bestimmung des linearen Ausgleichspolynoms $\varphi(x) = a_0 + a_1 x$.

```
disp('Test: Interpolation, polyfit, Haarsches System, Normal-GS')

% Referenz
x = 1:5;
y = [ 1 3 1 -1 -1 ];
n = max(size(x))-1;

plot(x,y,'yo',x,0.*y,'w',x,y,'y')
title('Referenz, Polygonzug und Interpolationspolynom')
hold on
% Interpolation
koeff = polyfit(x,y,n)
c = polyval(koeff,x)
xx = linspace(min(x),max(x),100);
```

```
yx = polyval(koeff,xx);
plot(xx,yx,'w')
print int7gr1.ps -dps
hold off

koeff =
   -0.0833    1.5000   -8.9167   19.5000  -11.0000

c =
    1.0000    3.0000    1.0000   -1.0000   -1.0000
```





Wie in der rechten Grpahik erkennbar, liefern die 3 Möglichkeiten die gleichen Ergebnisse. Unterschiede werden erst bei Vergrößerung der Anzahl der Basisfunktionen auftreten, wenn dann die zu lösenden LGS immer schlechtere Kondition bekommen.

```
% Approximation
disp('Test: polyfit, Haarsches System, Normal-GS')

disp('Variante 1')
koeff1 = polyfit(x,y,1)
y1 = polyval(koeff1,xx);

plot(x,y,'o',xx,0.*yx,'w',xx,yx,'w',xx,y1)
title('Interpolation (Grad=4) und Ausgleich (Grad=1)')
print int7gr2.ps -dps

disp('Variante 2')
H = [ones(size(x))' x']    % Haarsche Matrix
koeff2 = H\y'
r = H*koeff2-y'            % Residuen
norm(r)
y2 = polyval([koeff2(2),koeff2(1)],xx);
```

```
disp('Variante 3')
G = H'*H    % Matrix des Normalgleichungssystems
g = H'*y'
koeff3 = G\g
y3 = polyval([koeff3(2),koeff3(1)],xx);
```

Ergebnisse

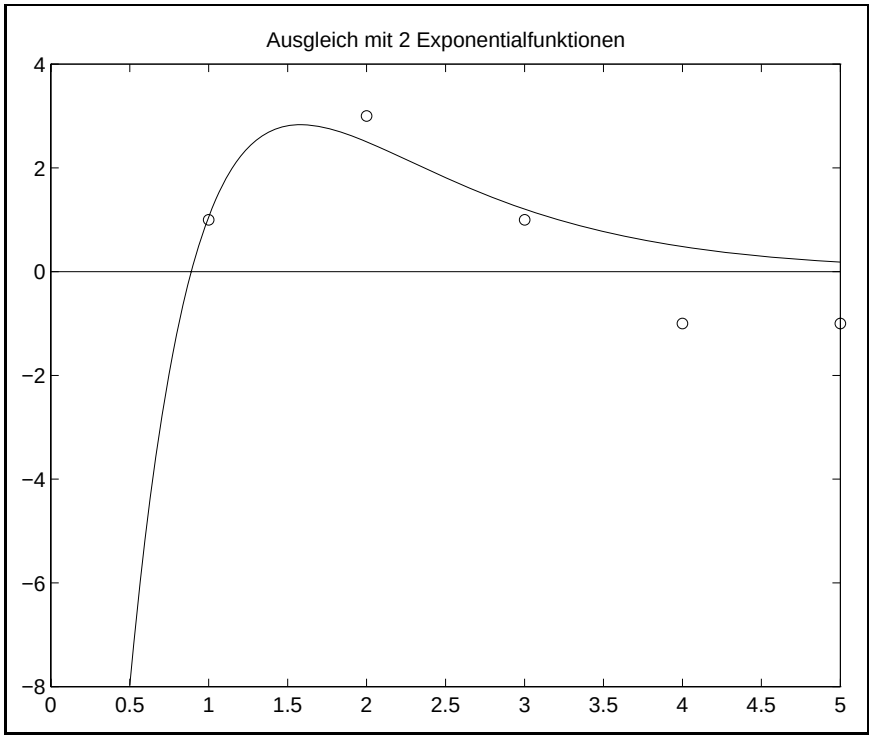
```
Variante 1
koeff1 =
   -0.8000    3.0000

Variante 2
H =
     1     1
     1     2
     1     3
     1     4
     1     5

koeff2 =
    3.0000
   -0.8000

r =
    1.2000
   -1.6000
   -0.4000
    0.8000
         0
ans =
    2.1909

Variante 3
G =
     5    15
    15    55

g =
     3
     1

koeff3 =
    3.0000
   -0.8000
```

Beispiel 2

Ausgleich der Daten aus Beispiel 1 durch die Funktion $\varphi(x) = c_1 e^{-\lambda_1 t} + c_2 e^{-\lambda_2 t}$, wobei die Parameter λ_i gegeben und die Koeffizienten c_i zu bestimmen sind.
Die Fehlerquadratabweichung (Norm) wird mittels einer "Zielfunktion" berechnet.

```
% zielf.m
function z = zielf(lam,y,t)
  z1 = [exp(-lam(1).*t) exp(-lam(2).*t)];
  z = norm(z1*(z1\y)-y);
```

```
disp('Ausgleich mit 2 Exponentialfunktionen')
t  = [ 1:5 ]';
yy = [ 1 3 1 -1 -1 ]';
lam= [ 1 2 ];

c  = [exp(-lam(1).*t) exp(-lam(2).*t)]\yy
xc = linspace(min(t)-0.5,max(t),100);
yc = c(1)*exp(-lam(1).*xc)+c(2)*exp(-lam(2).*xc);
plot(t,yy,'o',[0 xc],[0 0.*yc],'w',xc,yc)
title('Ausgleich mit 2 Exponentialfunktionen')
print int7gr3.ps -dps

% Abweichung als Zielfunktion
z = zielf(lam,yy,t);
s = sprintf('%6.4f',z);
disp(['Abw = ',s])
```



Ergebnisse

```
Ausgleich mit 2 Exponentialfunktionen

c =
   27.5810
  -67.1449

Abw = 1.9725
```

Beispiel 3

Gegeben sei die Referenz $\{(x_i, y_i),\ i = 0, 1, ..., N = 6\}$

x_i	0	1	2	3	4	5	6
y_i	2	0	-2	1	3	5	4

1. Bestimme das Newtonsche Interpolationspolynom höchstens 6.Grades zur Referenz.
2. Berechne die möglichen Ausgleichspolynome nach der Methode der kleinsten Quadrate mittels `polyfit`.
3. Wie verändert sich das Interpolationspolynom und die möglichen Ausgleichspolynome, wenn man noch den Punkt (7,1) hinzunimmt?

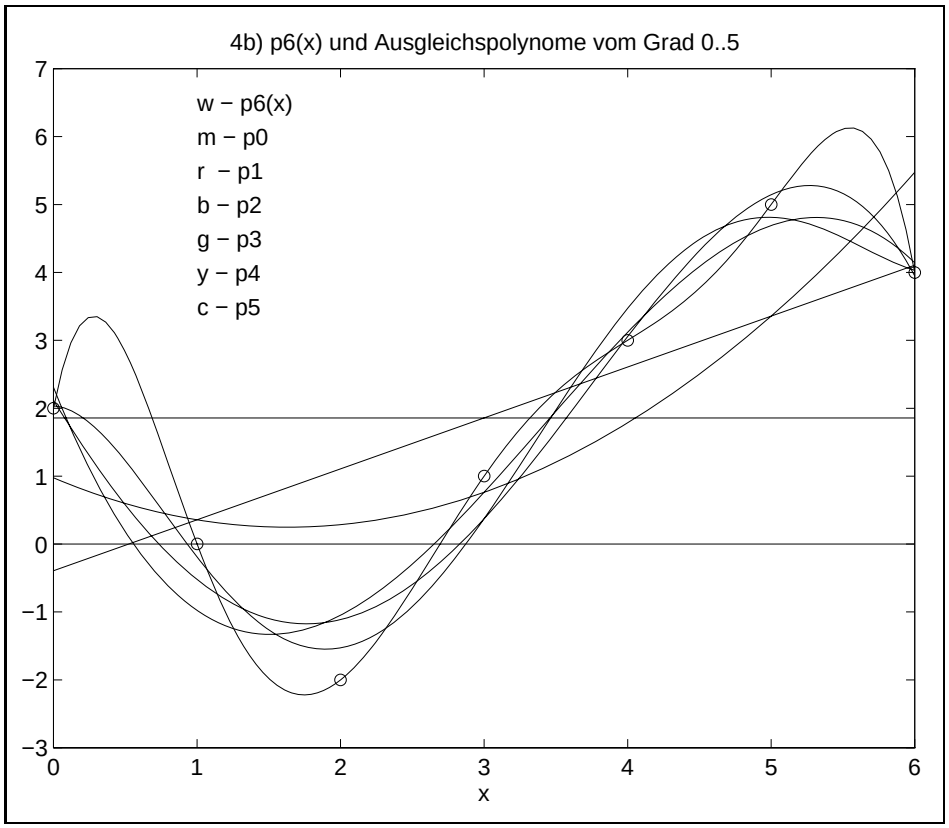
Die Ergebnisse sind numerisch und graphisch darzustellen.

```
% Referenz
x = 0:6;
y = [ 2  0  -2  1  3  5  4 ];
n = max(size(x))-1;

disp('Koeffizienten der NF des Interpolationspolynoms')
format long
a = polyfit(x,y,n)
format short
j = 1;
while (a(j)==0) & (j<n+1)
  j = j+1;
end;
neff = n+1-j

% Graphik
xi = linspace(min(x),max(x),100);
pxi = polyval(a,xi);
plot(x,y,'o',x,0*y,'w',xi,pxi,'y-')
title(['4a) Polynom ',sprintf('%1g',neff),'-ten Grades'])
```

```
xlabel('x')
text(3,2.9,['p',sprintf('%1g',neff),'(x)'])
print ser5gr04.ps -dps

disp('Ausgleichspolynome im Vergleich')
xi = linspace(min(x),max(x),100);
pxi = polyval(a,xi);
plot(x,y,'o',x,0.*y,'w',xi,pxi,'w-')
title(['4b) p6(x) und Ausgleichspolynome vom Grad 0..',sprintf('%1g',neff-1)])
xlabel('x')
text(1,6.5,'w - p6(x)')
text(1,6.0,'m - p0')
text(1,5.5,'r  - p1')
text(1,5.0,'b - p2')
text(1,4.5,'g - p3')
text(1,4.0,'y - p4')
text(1,3.5,'c - p5')
hold on
opt = ['m-'
       'r-'
       'b-'
       'g-'
       'y-'
       'c-' ];
for k=0:neff-1
  aa = polyfit(x,y,k);
  pxa = polyval(aa,xi);
  plot(xi,pxa,opt(k+1))
end
hold off
print ser5gr05.ps -dps
```

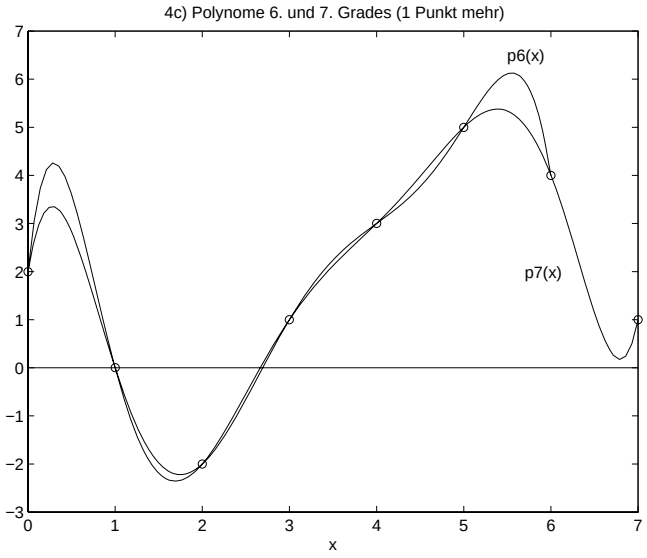
Ergebnisse

```
Koeffizienten der NF des Interpolationspolynoms

a =
  -0.04027777777778   0.75416666666663  -5.38194444444421  17.89583333333270
 -26.07777777777720  10.85000000000035   1.99999999999943

neff =
     6
```




```
% Zusaetzlicher Punkt (7,1) zur Referenz
xx = [x, 7];
yy = [y, 1];
nn = max(size(xx))-1;

disp('Koeffizienten der NF des Interpolationspolynoms')
format long
aa = polyfit(xx,yy,nn)
format short
j = 1;
while (aa(j)==0) & (j<nn+1)
  j = j+1;
end;
nneff = nn+1-j

% Graphik
xi1 = linspace(min(xx),max(xx),100);
pxi1 = polyval(aa,xi1);
plot(xx,yy,'o',xx,0.*yy,'w',xi,pxi,'y-',xi1,pxi1,'w-')
title(['4c) Polynome ',sprintf('%1g. und ',neff),...
sprintf('%1g',nneff),'. Grades (1 Punkt mehr)'])
xlabel('x')
```

```
text(5.5,6.5,['p',sprintf('%1g',neff),'(x)'])
text(5.7,2.0,['p',sprintf('%1g',nneff),'(x)'])
print ser5gr06.ps -dps
```

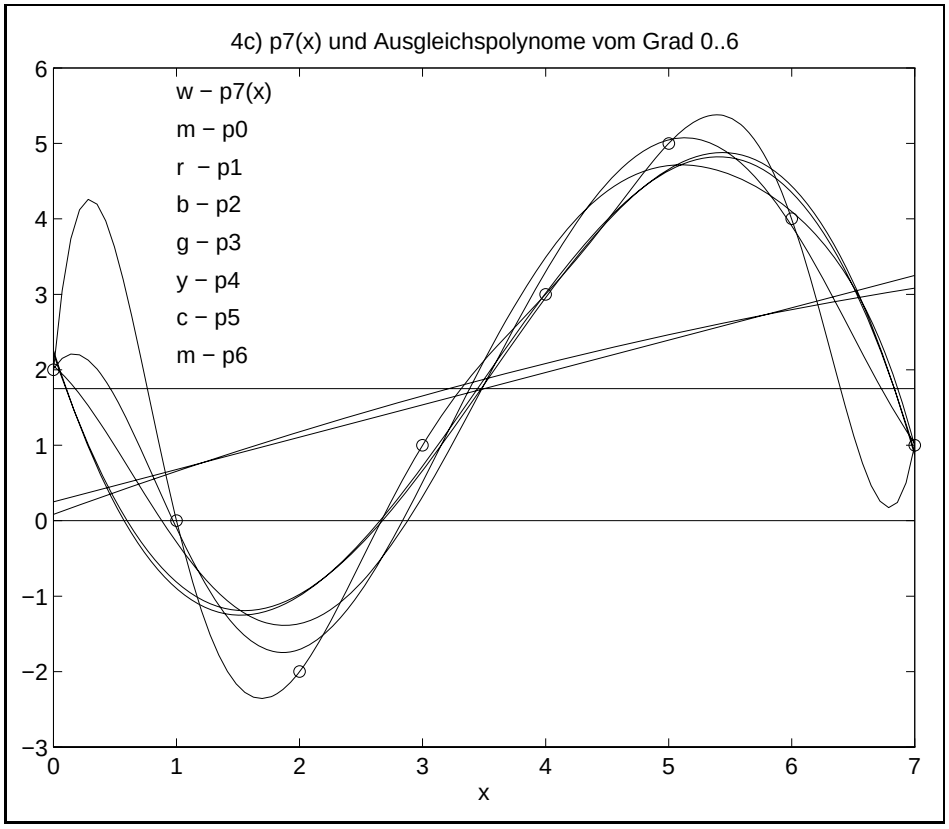



```
disp('Ausgleichspolynome im Vergleich')
xi1 = linspace(min(xx),max(xx),100);
pxi1 = polyval(aa,xi1);
plot(xx,yy,'o',xx,0.*yy,'w',xi1,pxi1,'w-')
title(['4c) p7(x) und Ausgleichspolynome vom Grad 0..',sprintf('%1g',nneff-1)])
xlabel('x')

text(1,5.7,'w - p7(x)')
text(1,5.2,'m - p0')
text(1,4.7,'r  - p1')
text(1,4.2,'b - p2')
text(1,3.7,'g - p3')
text(1,3.2,'y - p4')
text(1,2.7,'c - p5')
text(1,2.2,'m - p6')
hold on

opt1 = ['m-'
        'r-'
        'b-'
        'g-'
        'y-'
        'c-'
        'm-' ];
```

```
for k=0:nneff-1
  aa = polyfit(xx,yy,k);
  pxa1 = polyval(aa,xi1);
  plot(xi1,pxa1,opt1(k+1))
end
hold off
print ser5gr07.ps -dps
```

Ergebnisse

```
Koeffizienten der NF des Interpolationspolynoms

aa =
   0.00952380952381  -0.24027777777774   2.42083333333294 -12.38194444444235
  33.36249999999329 -42.87777777776418  17.70714285712625   2.00000000000930

nneff =
     7
```



Beispiel 4

Gegeben sei die Referenz $\{(x_i, y_i),\ i = 0, 1, ..., N = 12\}$

x_i	1	2	3	4	5	6	7	8	9	10	11	12	13
y_i	6	4	4	5	4	2	3	5	7	6	4	4	5

Bestimme die Ausgleichsgerade und untersuche die Abweichungen an den Stützstellen.

```
disp('Datenausgleich mit linearer Ansatzfunktion')
t = [1:13]';
y = [6 4 4 5 4 2 3 5 7 6 4 4 5]';
disp('Polynome p(1)t^m+...+p(m)t+p(m+1) vom Grad <=m')
m = 1
p = polyfit(t,y,m)
f = polyval(p,t);
disp('Tabelle')
disp('    t         y         f       r=y-f=y-Ac')
disp([t y f y-f])
no = norm(y-[t ones(size(t))]*([t ones(size(t))]\y),2)
```

Ergebnisse

```
Datenausgleich mit linearer Ansatzfunktion
Polynome p(1)t^m+...+p(m)t+p(m+1) vom Grad <=m

m = 1
p = 0.0330    4.3077

Tabelle
    t         y         f       r=y-f=y-Ac
    1.0000    6.0000    4.3407    1.6593
    2.0000    4.0000    4.3736   -0.3736
    3.0000    4.0000    4.4066   -0.4066
    4.0000    5.0000    4.4396    0.5604
    5.0000    4.0000    4.4725   -0.4725
    6.0000    2.0000    4.5055   -2.5055
    7.0000    3.0000    4.5385   -1.5385
    8.0000    5.0000    4.5714    0.4286
    9.0000    7.0000    4.6044    2.3956
   10.0000    6.0000    4.6374    1.3626
   11.0000    4.0000    4.6703   -0.6703
   12.0000    4.0000    4.7033   -0.7033
   13.0000    5.0000    4.7363    0.2637

no = 4.5862
```

Beispiel 5

Gegeben sind die $N+1$ Meßwerte

$t_j[min]$	7	12	17	22	27	32	37
$s_j[\%]$	83.7	72.9	63.2	54.7	47.5	41.4	36.3

für den Anteil s_j in Prozent eines Stoffes, der bei einer chemischen Reaktion nach t_j Minuten vom Beginn der Reaktion an noch im Versuchssystem verblieben ist.

1. Bestimme das Interpolationspolynom $p_6(t)$ höchstens 6.Grades zu den Meßpunkten.
2. Es wird ein quadratischer Zusammenhang vermutet. Berechne das Ausgleichspolynom $s_2(t)$ höchsten 2.Grades nach der Methode der kleinsten Quadrate.
3. Ermittle die Abweichungen

$$d_s = \sqrt{\sum_{j=0}^{N} [s_2(t_j) - s_j]^2} \quad \text{bzw.} \quad d_p = \sqrt{\sum_{j=0}^{N} [p_6(t_j) - s_j]^2}.$$

Die Ergebnisse sind numerisch und graphisch darzustellen.

```
% Referenz
n = 7;
t = [7, 12, 17, 22, 27, 32, 37];
s = [83.7, 72.9, 63.2, 54.7, 47.5, 41.4, 36.3];

plot(t,s,'o',t,s,'r',[min(t),max(t)],[0,0],'w')
title('4) Referenz, Polygonzug und p6(t)')
xlabel('t')
ylabel('s')
hold on

disp('a) Interpolation')
tt = linspace(min(t),max(t),100);
% Polynom 6.Grades
nn = 6
format long
p = polyfit(t,s,nn)
format short

% Darstellung als Polynom in Normalform:
% pn(t)=a[0]*t^n+a[1]*t^(n-1)+...+a[n]
% Achtung: Koeffizienten sind numerische Groessen
sh = [];
```

```
for k=0:nn
  if k<nn
    s1 = sprintf(' %11.9ft^%1g',abs(p(k+1)),nn-k);
   else
    s1 = sprintf(' %6.2f',abs(p(k+1)));
  end;
  if (p(k+1)<0)
    s1 = [' -',s1];
   elseif k>0
    s1 = [' +',s1];
  end;
  sh = [sh,s1];
end;

disp(['Normalform  pn(t) = ',sh])
ps = polyval(p,tt);
plot(tt,ps,[min(t),max(t)],[0,0],'w')
print ser6gr11.ps -dps
hold off

disp('b1) Ausgleich mit Methode der kleinsten Quadrate')
m = 2
p2 = polyfit(t,s,m)

disp('Darstellung in Normalform')
disp(['p2(t) = ',sprintf('%6.4f',p2(1)),'*t^2',...
     sprintf('%6.4f',p2(2)),'*t+',...
     sprintf('%6.4f',p2(3))])

polyval(p2,t)      % Kontrolle der Messwerte
p2s = polyval(p2,tt);

plot(t,s,'o',[min(t),max(t)],[0,0],'w',tt,p2s)
title('4b) Referenz und Ausgleichspolynom p2(t)')
xlabel('t')
ylabel('s')
print ser6gr12.ps -dps

disp('b2) Ausgleich mit LGS')
% y(t) = a0+a1*t+a2*t^2
A = [ones(size(t')) t' t'.*t']
a = A\s'

disp('c) Fehler')
disp('Fehler der Polynominterpolation an den Stuetzstellen ist 0.')
```

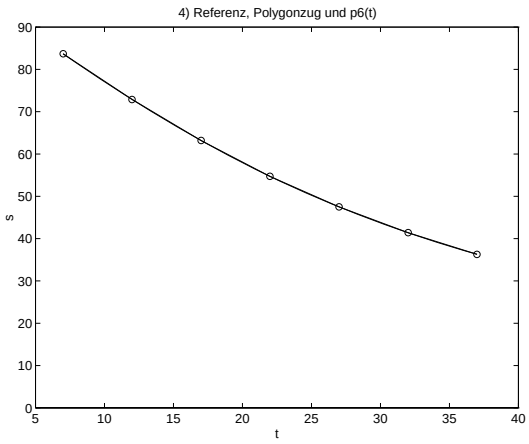
```
r = s'- A*a
no = norm(r,2)
disp('y(t) = a0+a1*t+a2*t^2 ')
disp(['       a(0..2) = ',sprintf('%9.4f',a)])
disp(' ')
disp('Fehler bei quadratischem Ausgleich')
disp(['norm = ',sprintf('%11.4e',no)])
```

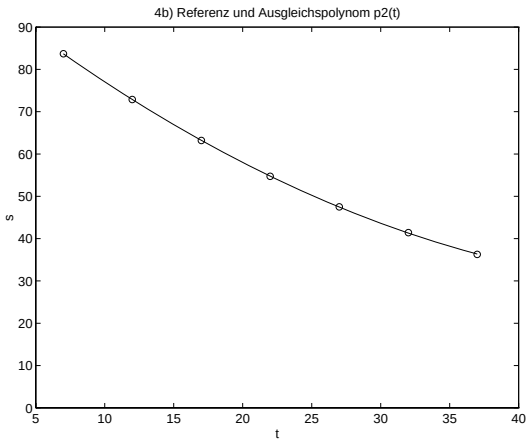
Ergebnisse

```
a) Interpolation
nn =
     6

p =
  1.0e+002 *
   0.00000000062222 -0.00000008080000  0.00000409288889 -0.00010244400000
   0.00155930488889 -0.03446585280001  1.02845109760005

Normalform  pn(t) =  0.000000062t^6-0.000008080t^5+0.000409289t^4
                   - 0.010244400t^3+0.155930489t^2-3.446585280t^1+102.85

b1) Ausgleich mit Methode der kleinsten Quadrate
m =
     2

p2 =
    0.0234   -2.6066  100.7911

Darstellung in Normalform
p2(t) = 0.0234*t^2-2.6066*t+100.7911

ans =
   83.6905   72.8786   63.2357   54.7619   47.4571   41.3214   36.3548

b2) Ausgleich mit LGS

A =
           1           7          49
           1          12         144
           1          17         289
           1          22         484
           1          27         729
           1          32        1024
           1          37        1369
```

```
a =
  100.7911
   -2.6066
    0.0234

c) Fehler

Fehler der Polynominterpolation an den Stuetzstellen ist 0.

r =
    0.0095
    0.0214
   -0.0357
   -0.0619
    0.0429
    0.0786
   -0.0548

no =
    0.1291

y(t) = a0+a1*t+a2*t^2
       a(0..2) =  100.7911  -2.6066   0.0234

Fehler bei quadratischem Ausgleich
norm = 1.2910e-001
```

Die sehr geringen Abweichungen sind in groben graphischen Auswertungen nicht zu erkennen. Dafür müßten lokale Ausschnitte vergrößert werden.





1.2.4 Nichtlinearer Ausgleich

Algorithmus

1. Wahl eines geeigneten nichtlinearen Ansatzes $y = \varphi(x)$.
2. Mittels Koordinatentransformation

$$X = X(x, y), \quad Y = Y(x, y)$$

Überführung in einen linearen oder quadratischen Ansatz.

$$\begin{aligned} Y &= \psi(X) = A_0 + A_1 X && \text{(linearer Ausgleich)} \\ Y &= \psi(X) = A_0 + A_1 X + A_2 X^2 && \text{(quadratischer Ausgleich)} \end{aligned}$$

3. Polynomausgleich liefert die Koeffizienten A_0, A_1, A_2.
4. Rücktransformation von $Y = \psi(X)$ in die Ausgangskoordinaten (x, y), d.h. Berechnung der Parameter in $\varphi(x)$ aus A_k.

Ansatz 1 Potenzfunktion $y = \varphi(x) = ax^b$

$$\begin{aligned} \ln y &= \ln a + b \ln x \\ Y &= A_0 + A_1 X \end{aligned}$$

mit $X = \ln x$, $Y = \ln y$ und Voraussetzung $x_j, y_j > 0$, $j = 0(1)N$.
Rücktransformation $a = e^{A_0}$, $b = A_1$.

Ansatz 2 Exponentialfunktion $y = \varphi(x) = ae^{bx}$

$$\begin{aligned} \ln y &= \ln a + bx \\ Y &= A_0 + A_1 X \end{aligned}$$

mit $X = x$, $Y = \ln y$ und Voraussetzung $y_j > 0$, $j = 0(1)N$.
Rücktransformation $a = e^{A_0}$, $b = A_1$.

Ansatz 3 Hyperbelfunktion $y = \varphi(x) = a + \frac{b}{x}$

$$\begin{aligned} y &= a + b\frac{1}{x} \\ Y &= A_0 + A_1 X \end{aligned}$$

mit $X = \frac{1}{x}$, $Y = y$ und Voraussetzung $x_j \neq 0$, $j = 0(1)N$.
Rücktransformation $a = A_0$, $b = A_1$.

oder

$$\begin{aligned} xy &= b + ax \\ Y &= A_0 + A_1 X \end{aligned}$$

mit $X = x,\ \ Y = xy$ und Voraussetzung $x_j \neq 0,\ j = 0(1)N.$
Rücktransformation $\quad b = A_0,\ \ a = A_1$

Ansatz 4 Törnquistfunktion $y = \varphi(x) = \dfrac{ax}{x+b}$

$$\begin{aligned} \frac{1}{y} &= \frac{1}{a} + \frac{b}{a}\frac{1}{x} \\ Y &= A_0 + A_1 X \end{aligned}$$

mit $X = \frac{1}{x},\ \ Y = \frac{1}{y}$ und Vorauss. $x_j, y_j > 0,\ j = 0(1)N.$
Rücktransformation $\quad a = \frac{1}{A_0},\ \ b = \frac{A_1}{A_0}.$

Ansatz 5 Exponentialfunktion $y = \varphi(x) = ae^{bx+cx^2}$

$$\begin{aligned} \ln y &= \ln a + bx + cx^2 \\ Y &= A_0 + A_1 X + A_2 X^2 \end{aligned}$$

mit $X = x,\ \ Y = \ln y$ und Voraussetzung $y_j > 0,\ j = 0(1)N.$
Rücktransformation $\quad a = e^{A_0},\ \ b = A_1,\ \ c = A_2.$

Ansatz 6 Allgemeine Hyperbelfunktion $y = \varphi(x) = \dfrac{1}{a_0 + a_1 x + ... + a_n x^n}$

$$\begin{aligned} \frac{1}{y} &= a_0 + a_1 x + ... + a_n x^n \\ Y &= A_0 + A_1 X + ... + A_n X^n \end{aligned}$$

mit $X = x,\ \ Y = \frac{1}{y}$ und Voraussetzung $y_j > 0,\ j = 0(1)N.$
Rücktransformation $\quad a_0 = A_0,\ a_1 = A_1, ...,\ a_n = A_n.$

Die 4 ersten Ansätze ($m = -1, -2, -3, -4$) sind neben dem Polynomausgleich in folgender MATLAB Funktion enthalten. Dazu wird noch für graphische Auswertungen eine Referenz mit den Argumenten xi verarbeitet und die Abweichung

$$Error = \|\varphi - y\|_2^2 = \sum_{j=0}^{N} [\varphi(x_j) - y_j]^2$$

berechnet.

```
% APPROX.M
% Ausgleichsrechnung mit verschiedenen
% polynomialen (m>0) und nichtlinearen (m<0) Ansaetzen fuer Daten (x,y)
% Berechnung von Funktionswert fuer Argument xi und Fehler
%
function  [yi,Error] = approx(x,y,xi,m)

% Polynom vom Grad m
   if (m > 0)
      p = polyfit(x,y,m)
      yi = polyval(p,xi);
      si = polyval(p,x);
      Error = norm(si-y,2)^2;
   end;

% Potenzfunktion
   if (m==-1)
      X = log(x);
      Y = log(y);
      p = polyfit(X,Y,1);
      a = exp(p(2))
      b = p(1)
      yi = a.*(xi.^b);
      si = a.*(x.^b);
      Error = norm(si-y,2)^2;
   end;

% Exponentialfunktion
   if (m==-2)
      X = x;
      Y = log(y);
      p = polyfit(X,Y,1);
      a = exp(p(2))
      b = p(1)
      yi = a.*exp(xi.*b);
      si = a.*exp(x.*b);
      Error = norm(si-y,2)^2;
   end;

% Hyperbelfunktion
   if (m==-3)
      X = 1./x;
      Y = y;
      p = polyfit(X,Y,1);
      a = p(2)
```

```
      b = p(1)
      yi = a + b./xi;
      si = a + b./x;
      Error = norm(si-y,2)^2;
   end;

% Toernquistfunktion
   if (m==-4)
      X = 1./x;
      Y = 1./y;
      p = polyfit(X,Y,1);
      a = 1/p(2)
      b = p(1)/p(2)
      yi = (a*xi)./(b + xi);
      si = (a*x )./(b + x );
      Error = norm(si-y,2)^2;
   end;
% Ende function approx
```

Beispiel

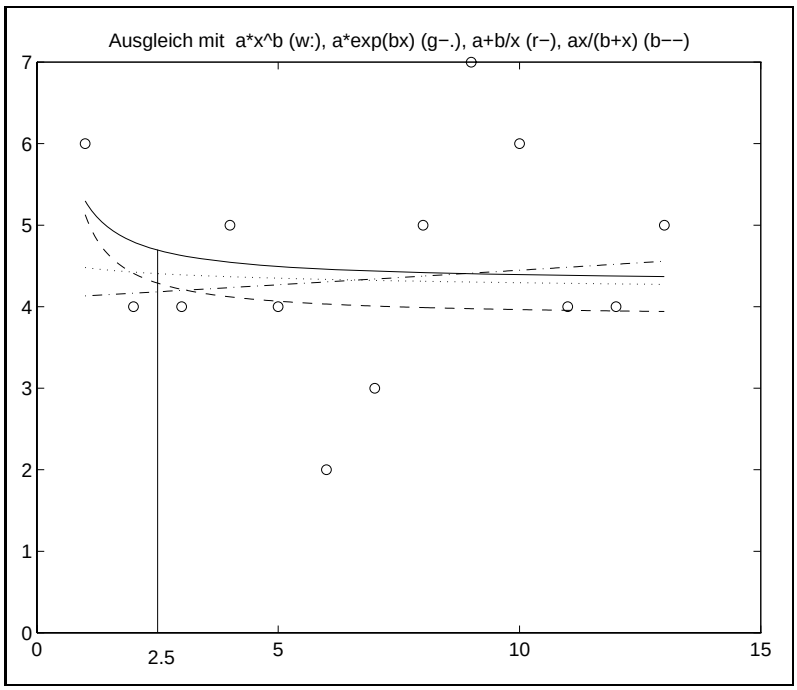
```
disp('Datenausgleich mit verschiedenen Ansatzfunktionen')
x = [1:13]';
y = [6 4 4 5 4 2 3 5 7 6 4 4 5]';
xi = 2.5
xp = 1:0.05:13;
plot(x,y,'o')
title('Datenpunkte P(x,y)')
disp('Ausgleichspolynom vom Grad 1')
[yi0,Error0] = approx(x,y,xi,1);
disp(['yi0 = ',sprintf('%6.4f',yi0),'  Error0 = ',sprintf('%11.4e',Error0)])
[yp,Error0] = approx(x,y,xp,1);
plot(x,y,'o',xp,yp,'-')
title('Datenpunkte und Ausgleich mit Geraden')
axis([0 15 0 7])

disp('Nichtlineare Ausgleichsfunktionen')
[yi1,Error1] = approx(x,y,xi,-1);
disp(['yi1 = ',sprintf('%6.4f',yi1),'  Error1 = ',sprintf('%11.4e',Error1)])
[yp,Error1] = approx(x,y,xp,-1);
plot(x,y,'o',[2.5 2.5],[0 4.7],'w-'xp,yp,'w:')
title('Ausgleich mit  a*x^b (w:), a*exp(bx) (g-.), a+b/x (r-), ax/(x+b) (b--)')
text(2.3,-0.3,['2.5'])
axis([0 15 0 7])
hold on
```

```
[yi2,Error2] = approx(x,y,xi,-2);
disp(['yi2 = ',sprintf('%6.4f',yi2),'  Error2 = ',sprintf('%11.4e',Error2)])
[yp,Error0] = approx(x,y,xp,-2);
plot(xp,yp,'g.')

[yi3,Error3] = approx(x,y,xi,-3);
disp(['yi3 = ',sprintf('%6.4f',yi3),'  Error3 = ',sprintf('%11.4e',Error3)])
[yp,Error4] = approx(x,y,xp,-3);
plot(xp,yp,'r-')

[yi4,Error4] = approx(x,y,xi,-4);
disp(['yi4 = ',sprintf('%6.4f',yi4),'  Error4 = ',sprintf('%11.4e',Error4)])
[yp,Error4] = approx(x,y,xp,-4);
plot(xp,yp,'b--')

print ausgl1g1.ps -dps
hold off
```

Ausgewählte Ergebnisse

```
Datenausgleich mit verschiedenen Ansatzfunktionen

xi = 2.5000

Ausgleichspolynom vom Grad 1
p =  0.0330    4.3077
yi0 = 4.3901  Error0 = 2.1033e+001

Nichtlineare Ausgleichsfunktionen
a =   4.4812
b =  -0.0185
yi1 = 4.4058  Error1 = 2.1711e+001

a =   4.0981
b =   0.0082
yi2 = 4.1827  Error2 = 2.1524e+001

a =   4.2927
b =   1.0047
yi3 = 4.6946  Error3 = 2.0430e+001

a =   3.8669
b =  -0.2461
yi4 = 4.2890  Error4 = 2.2332e+001
```

Die Hyperbelfunktion (solid line) hat die geringste Abweichung.

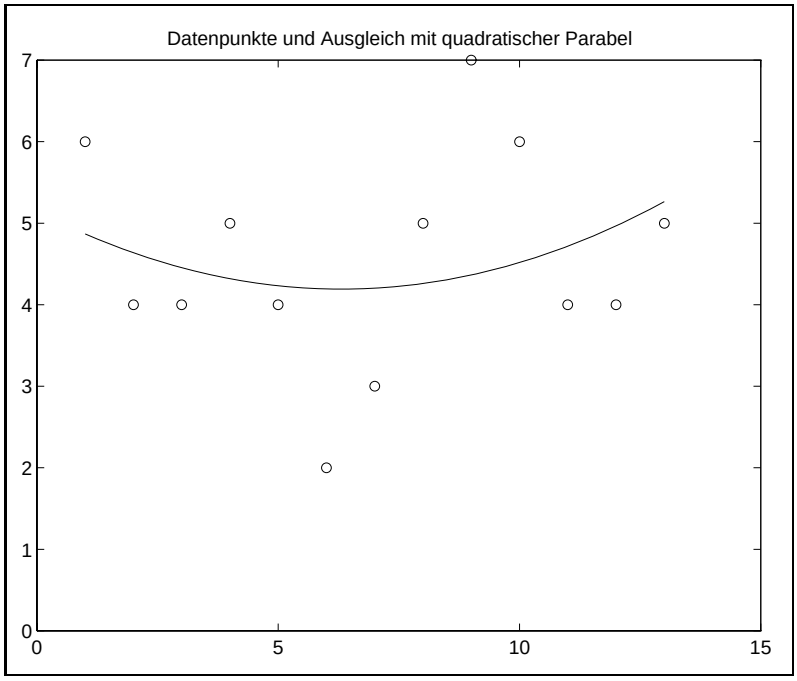


Beim folgenden Ausgleich mittels Polynom 2.Grades wird zusätzlich die Wirkungsweise der MATLAB Funktionen `poly2sym, eval, subs, numeric` getestet.

```
% Ausgleichspolynome vom Grad m
m = 2
p = polyfit(x,y,m)
% Darstellung in Normalform
disp(['p2(x) = ',sprintf('%6.4f',p(1)),'*x^2',...
     sprintf('%6.4f',p(2)),'*x+',sprintf('%6.4f',p(3))])
% Analoge Darstellung in NF mittels
poly2sym(p,'x')
poly2sym(p,'z')
z=1
eval(poly2sym(p,'z'))
subs(poly2sym(p),1,'x')
numeric(subs(poly2sym(p),1,'x'))

f = polyval(p,x);
f = f'
xx = 1:0.1:13;
ff = polyval(p,xx);
plot(x,y,'o',xx,ff,'-')
title('Datenpunkte und Ausgleich mit quadratischer Parabel')
axis([0 15 0 7])
print ausgl1g2.ps -dps
```

Ausgewählte Ergebnisse

```
m =  2

p =  0.0240   -0.3027    5.1469

p2(x) = 0.0240*x^2-0.3027*x+5.1469

ans =
  1727654602307963/72057594037927936*x^2
 -1363227459633627/4503599627370496*x+736/143

ans =
  1727654602307963/72057594037927936*z^2
 -1363227459633627/4503599627370496*z+736/143

z =  1

ans =  4.8681
ans =  50162379392403261029/10304235947423694848
ans =  4.8681

f =  4.8681  4.6374  4.4545  4.3197  4.2328  4.1938  4.2028
     4.2597  4.3646  4.5175  4.7183  4.9670  5.2637
```



1.2.5 Ausgleich durch lineare Funktionen in $\mathbb{R}^n$

Aufgabenstellung

- Grundmenge $D \subset \mathbb{R}^n$ sowie Meßpunkte $x_j = (x_{j1}, x_{j2}, ..., x_{jn}) \in D$ und Meßwerte $y_j,\ j = 0(1)N$.
- Ansatzfunktion (Hyperebene) $\varphi(x)$ der Form

$$\varphi(x) = \varphi(x_1, x_2, ..., x_n) = a_0 + a_1x_1 + a_2x_2 + ... + a_nx_n$$

  mit $x = (x_1, x_2, ..., x_n)$.
- Spezialfall $n = 2$ (Ebenenausgleich)

$$\varphi(x_1, x_2) = a_0 + a_1x_1 + a_2x_2.$$

Verallgemeinerte HAARsche Bedingung

Mit den $n + 1$ Basisfunktionen

$$\varphi_0(x) = 1, \quad \varphi_i(x) = x_i,\ i = 1(1)n,$$

lautet die Matrix

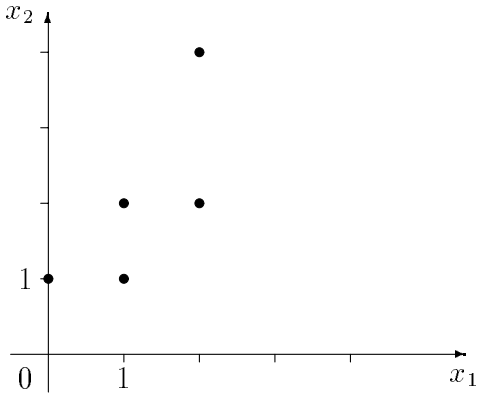
$$\Phi = \begin{pmatrix} 1 & x_{01} & x_{02} & \cdots & x_{0n} \\ 1 & x_{11} & x_{12} & \cdots & x_{1n} \\ 1 & x_{21} & x_{22} & \cdots & x_{2n} \\ & \cdots\cdots\cdots\cdots\cdots & & & \\ 1 & x_{N1} & x_{N2} & \cdots & x_{Nn} \end{pmatrix},$$

die nicht notwendig den maximalen Rang $r = \min(n, N) + 1$ besitzt.

Beispiel

Zu folgenden Meßwerten

x_1	0	1	1	2	2
x_2	1	1	2	4	2
y	2	3	4	4	3



ist eine lineare Ausgleichsfunktion (Ausgleichsebene)

$$y = \varphi(x_1, x_2) = a_0 + a_1x_1 + a_2x_2$$

zu ermitteln. Der Rang von

$$\Phi = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 4 \\ 1 & 2 & 2 \end{pmatrix}$$

ist $r = n + 1 = 3$ (spaltenregulär).

FALKsches Schema

Tableau zur günstigen Berechnung der Skalarprodukte und zum Aufstellen des Normalgleichungssystems

					1	x_1	x_2	y
					1	2	2	3
					1	2	4	4
					1	1	2	4
					1	1	1	3
					1	0	1	2
1	1	1	1	1	5	6	10	16
2	2	1	1	0	6	10	15	21
2	4	2	1	1	10	15	26	35

Normalgleichungssystem

$$\begin{aligned} 5a_0 + 6a_1 + 10a_2 &= 16 \\ 6a_0 + 10a_1 + 15a_2 &= 21 \\ 10a_0 + 15a_1 + 26a_2 &= 35 \end{aligned}$$

Ausgleichsfunktion $y = \varphi(x) = \frac{28}{13} + \frac{3}{13}x_1 + \frac{5}{13}x_2$

Ausgewählte Ergebnisse von MATLAB Anweisungen

```
koeff =
    2.1538
    0.2308
    0.3846

residuen =
    0.5385
   -0.2308
   -0.8462
    0.1538
    0.3846

norm =
    1.1094

2.1538+0.2308*x(1)+0.3846*x(2)
```

1.3 Übersicht zu Arten der Approximation

Dabei geht es um die Approximation in linearen normierten Räumen mit den Basisfunktionen $\varphi_i(x),\ i = 0, 1, ..., n$.

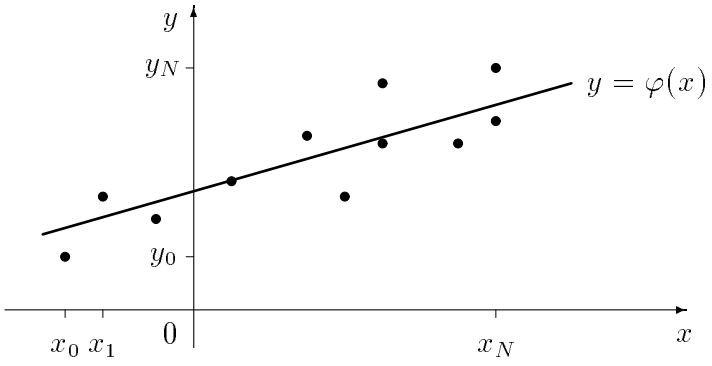
Die approximierende Funktion mit den freien Parametern a_i hat die Gestalt

$$\varphi(x) = \sum_{i=0}^{n} a_i \varphi_i(x).$$

	Diskrete Approximation	**Kontinuierliche/stetige Approximation**
Approximation im Mittel	$\sum_{j=0}^{N} (y_j - \varphi(x_j))^2 \longrightarrow \min$ Gewicht=1	$\int_a^b (f(x) - \varphi(x))^2 dx \longrightarrow \min$
Verfahren	Gaußsche Methode der kleinsten Quadrate	Numerische Fourieranalyse, Fast Fourier Transformation
Gleichmäßige Approximation	$\max_{j=0(1)N} \lvert y_j - \varphi(x_j)\rvert \longrightarrow \min$	$\max_{a \le x \le b} \lvert f(x) - \varphi(x)\rvert \longrightarrow \min$
Verfahren	Austauschalgorithmus von Stiefel, Simplex-Algorithmus	Remes-Algorithmus, Telescoping

1.3.1 Approximation im Mittel

- Diskrete Approximation im Mittel, Ausgleichsrechnung
  Referenz aus hinreichend vielen Punkten $(x_j, y_j),\ j = 0, 1, ..., N$



$$F(a_0, a_1, ..., a_n) = \sum_{j=0}^{N} \omega(x_j)(y_j - \varphi(x_j))^2 \ \rightarrow \ \min,$$

$\omega(x) > 0$ Gewichtsfunktion.

Bestimmung der freien Parameter a_i mittels notwendiger Bedingung am Minimum.

$$\frac{\partial F}{\partial a_i} = 0, \quad i = 0, 1, ..., n.$$

- Kontinuierliche Approximation im Mittel
  Beste Approximation mit orthogonalen Funktionensystemen

  Funktionenraum $L_{2,\omega}[a, b]$ mit Skalarprodukt $(f, g) = \int_a^b \omega(x) f(x) g(x)\, dx,$

  Norm $\|f\|^2_{L_{2,\omega}} = (f, f) = \int_a^b \omega(x)(f(x))^2\, dx$

  (a)

$$F(a_0, a_1, ..., a_n) = \|f - \varphi\|^2 = \int_a^b \omega(x)(f(x) - \varphi(x))^2\, dx \ \rightarrow \ \min$$

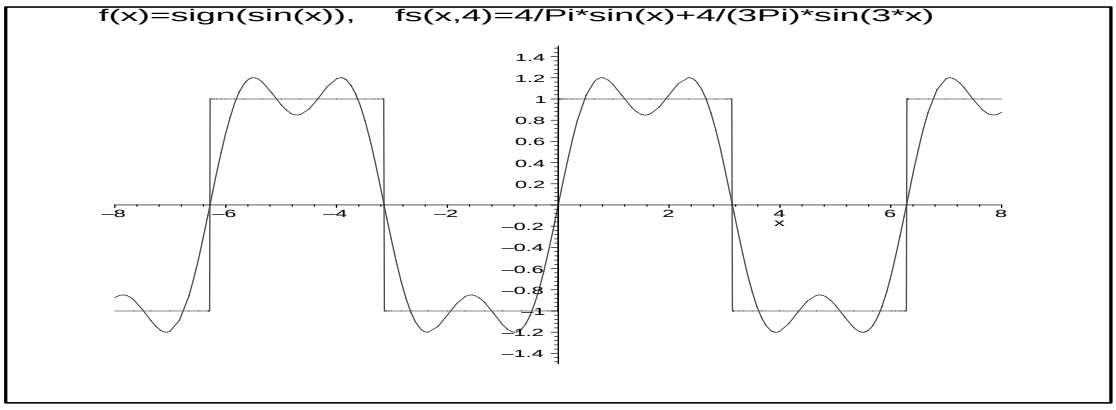
  Bestimmung der freien Parameter a_i mittels notwendiger Bedingung am Minimum.

$$\frac{\partial F}{\partial a_i} = 0, \quad i = 0, 1, ..., n.$$

  (b) Entwicklung nach orthogonalen Funktionensystemen, Fourierreihen, Orthonormalsystem (ONS) $\varphi_i(x)$

$$\|f - \varphi\|^2 = (f - \varphi, f - \varphi) \ \rightarrow \ \min$$

  Diskrete Fouriertransformation (DFT) für Signalfunktion mit dem Orthogonalsystem (OGS) $\{1, \sin(kx), \cos(kx)\}$

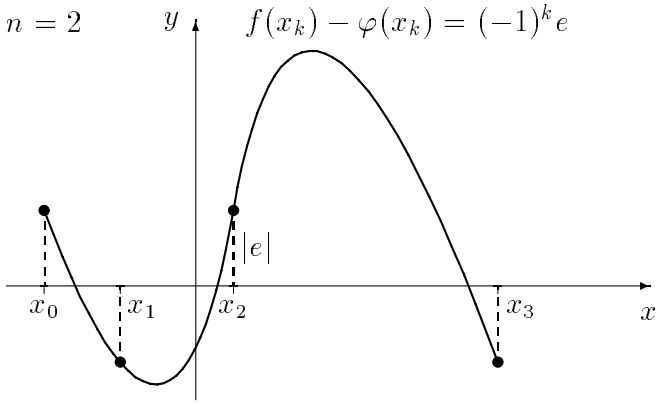


1.3.2 Gleichmäßige Approximation

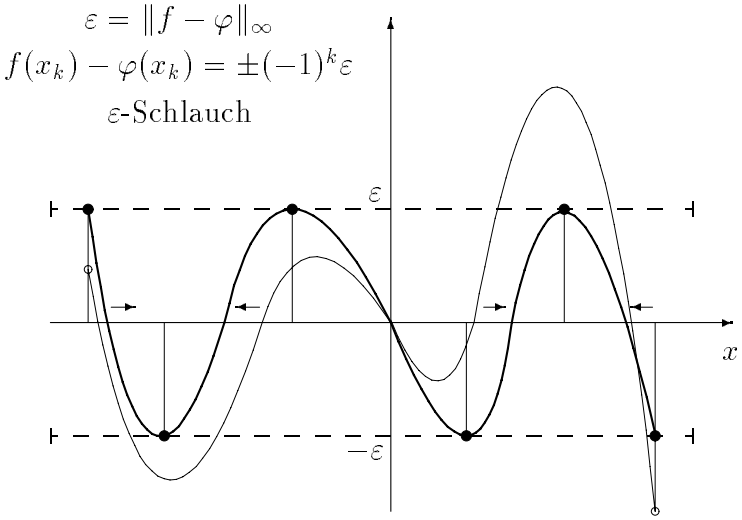
- Polynom bester Approximation auf endlicher Menge
  $X = \{x_0, x_1, ..., x_{n+1}\} \subset [a,b],\ x_k$ verschieden.

$$f(x_k) - \varphi(x_k) = (-1)^k e, \quad |e| = \max_{x_k \in X} |f(x_k) - \varphi(x_k)|.$$

  Bestimmung der freien Parameter a_i und e mittels LGS oder Schema der dividierten Differenzen.



- Verallgemeinertes Polynom bester Approximation in $C[a,b]$
  (a) $a \le x_0 < x_1 < ... < x_{n+1} \le b$
  (b) $f(x_k) - \varphi(x_k) = (-1)^k \alpha \|f - \varphi\|_\infty, \quad \alpha = \pm 1$
  $\{x_0, x_1, ..., x_{n+1}\}$ ist dann die Tschebyscheff-Alternante.



2 Numerische Integration

2.1 Grundlagen

- **Problemstellung**

  Approximation des Riemannschen Integrals (bestimmtes Integral, Funktional)

$$I \;=\; I(f) \;=\; I_a^b(f) \;=\; \int_a^b f(x)\,dx, \quad -\infty < a < b < \infty, \quad f \in C[a,b]$$

- **Eigenschaften des Funktionals**

  I ist ein positives, lineares und beschränktes Funktional (damit stetiges) auf dem Raum $C[a,b]$ sowie additiv bez. der Intervallzerlegung.

  - Positivität

$$\text{Falls } f(x) \geq 0, \text{ dann } \; I(f) \geq 0.$$

  - Linearität

$$I(\alpha f + \beta g) = \alpha I(f) + \beta I(g), \quad f, g \in C[a,b],\ \alpha, \beta \in \mathbb{R}$$

  - Beschränkheit

$$|I(f)| \;\leq\; \int_a^b |f(x)|\,dx \;=\; I(|f|) \;\leq\; (b-a)\,\|f\|_\infty$$

  - Additivität bez. Intervallzerlegung

$$I_a^b(f) = I_a^c(f) + I_c^b(f), \quad c \in [a,b]$$

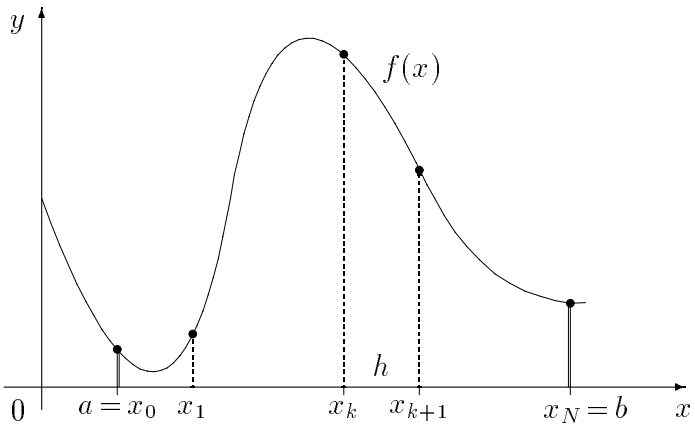
- **Kondition der Integralberechnung**

  Im Gegensatz zur Interpolation und numerischen Differentiation hat die numerische Intergration "glättende" Eigenschaften und ist damit ein gutartiges Problem.

- **Intervallzerlegung**

  N Teilintervalle : gleich große $\quad h = x_{k+1} - x_k = \dfrac{b-a}{N}$

  bzw. nicht uniform $\quad h_k = x_{k+1} - x_k$



- **Teilintegrale**
  Auf $[x_k, x_{k+1}]$ betrachtet man

$$I = \int_{x_k}^{x_{k+1}} f(x)\, dx$$

- **Transformation auf Standardintervall**
  $[-\frac{h}{2}, \frac{h}{2}]$, $[-h, h]$, $[0, h]$ oder $[0, 2h]$ (hier ersteres)

$$I = \int_{-h/2}^{h/2} f(x)\, dx$$

2.2 Integrationsformeln und ihre Eigenschaften

- **Einfache Integrationsformeln auf** $[a, b]$

(1) Rechteckregel (Links) : $R_L = (b-a)\, f(a)$

(2) Rechteckregel (Rechts) : $R_R = (b-a)\, f(b)$

(3) Rechteckregel (Mitte) : $R_M = (b-a)\, f(\frac{a+b}{2})$

(4) Trapezregel : $T = \frac{1}{2}(b-a)\, [f(a) + f(b)]$

(5) Simpsonregel : $S = \frac{1}{6}(b-a)\, [f(a) + 4f(\frac{a+b}{2}) + f(b)]$

- **Einfache Näherungsformeln auf Standardintervall** $[-\frac{h}{2}, \frac{h}{2}]$

(1) Rechteckregel (Links) : $R_L = h\ f(-\frac{h}{2})$

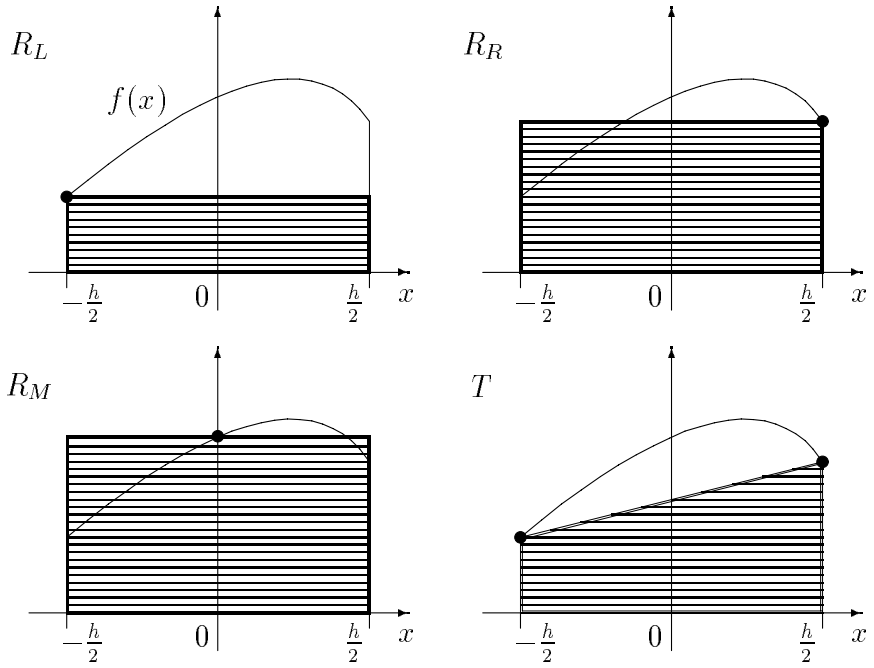
(2) Rechteckregel (Rechts) : $R_R = h\ f(\frac{h}{2})$

(3) Rechteckregel (Mitte) : $R_M = h\ f(0)$

(4) Trapezregel : $T = h\ [\frac{1}{2}f(-\frac{h}{2}) + \frac{1}{2}f(\frac{h}{2})]$
$= \frac{h}{2}\ [f(-\frac{h}{2}) + f(\frac{h}{2})]$

(5.1) Simpsonregel in $[-\frac{h}{2}, \frac{h}{2}]$: $S = S_h = h\ [\frac{1}{6}f(-\frac{h}{2}) + \frac{4}{6}f(0) + \frac{1}{6}f(\frac{h}{2})]$
$= \frac{h}{6}\ [f(-\frac{h}{2}) + 4f(0) + f(\frac{h}{2})]$

(5.2) Simpsonregel in $[-h, h]$: $S = S_{2h} = 2h\ [\frac{1}{6}f(-h) + \frac{4}{6}f(0) + \frac{1}{6}f(h)]$
$= \frac{h}{3}\ [f(-h) + 4f(0) + f(h)]$



- Eigenschaften der Integrationsformel

1. Eine *Integrationsformel (Integrationsregel, Quadraturformel)* mit $n+1$ Stützstellen zur Bestimmung des Integrals

$$I = \int_{-\frac{h}{2}}^{\frac{h}{2}} f(x)\ dx$$

besitzt die allgemeine Form

$$I_{n+1} \;=\; I_{n+1}(f) \;=\; h \sum_{k=0}^{n} A_k f(x_k) \;=\; \sum_{k=0}^{n} w_k f(x_k).$$

2. Die Werte x_k mit $-\frac{h}{2} \le x_0 < x_1 < x_2 < \ldots < x_n \le \frac{h}{2}$ heißen *Integrationsknoten (Knoten, Integrationsstützstellen)*, die Werte A_k heißen *Integrationsgewichte (Gewichte)*, die Werte $w_k = hA_k$ werden ebenfalls Gewichte oder einfach Koeffizienten genannt. Für die Positivität von I_{n+1} werden nichtnegative Gewichte gebraucht.
   Sinnvoll erweist sich, den Abstand $\bar{h}$ der (äquidistanten) Integrationsknoten einzuführen.

3. **Integrationsfehler**
   Die Differenz
   $$R_{n+1} = I - I_{n+1}$$
   heißt *Integrationsfehler (Quadraturfehler)*.
   Der Fehler hängt i.a. ab von der Glattheit des Integranden, der Genauigkeit der Quadraturformel sowie der Unterteilung des Integrationsintervalls in Teilintervalle.

4. **Konsistenz, Mindestanforderung**
   Um die Exaktheit für die Integration einer konstanten Funktion zu sichern, ist die eine natürliche Bedingung
   $$\sum_{k=0}^{n} A_k = 1 \quad \text{bzw.} \quad \sum_{k=0}^{n} w_k = h,$$
   die sich einfach aus der Gleichheit bei $f(x) = c = const$ ergibt.
   $$hc = \int_{-\frac{h}{2}}^{\frac{h}{2}} f(x)\, dx = h \sum_{k=0}^{n} A_k f(x_k) = hc \sum_{k=0}^{n} A_k$$
   Obige einfache Quadraturformeln erfüllen diese Konsistenzbedingung.

2.3 Berechnung der Knoten und Gewichte

Methode der unbestimmten Koeffizienten (Koeffizientenabgleich)

Annahme: f ist analytisch in $[a,b]$.

- Entwicklung von $f(x)$ an der Stelle $x^* = 0$ in eine Taylorreihe.

$$f(x) = \sum_{\nu=0}^{\infty} a_\nu x^\nu \quad \text{mit} \quad a_\nu = \frac{f^{(\nu)}(0)}{\nu!}$$

- Einsetzen der Taylorreihe in I und I_{n+1} liefert für den Integrationsfehler

$$\begin{aligned} R_{n+1} &= I - I_{n+1} \\ &= \int_{-\frac{h}{2}}^{\frac{h}{2}} f(x)\, dx - h \sum_{k=0}^{n} A_k f(x_k) \\ &= \int_{-\frac{h}{2}}^{\frac{h}{2}} \sum_{\nu=0}^{\infty} a_\nu x^\nu dx - h \sum_{k=0}^{n} A_k \sum_{\nu=0}^{\infty} a_\nu x_k^\nu \\ &= h \sum_{\nu=0}^{\infty} \left\{ \frac{1}{\nu+1} \left(\frac{h}{2}\right)^\nu \frac{1+(-1)^\nu}{2} - \sum_{k=0}^{n} A_k x_k^\nu \right\} a_\nu . \end{aligned}$$

- Bestimmungsgleichungen (G) für $x_k,\ A_k$

$$\boxed{\sum_{k=0}^{n} A_k\, x_k^\nu \;=\; \frac{1}{\nu+1} \left(\frac{h}{2}\right)^\nu \frac{1+(-1)^\nu}{2}, \quad \nu = 0(1)K.}$$

- Wichtige Formelgruppen
  Solche ergeben sich bei

$$K = K(n) = \begin{cases} n & \text{Newton-Cotes-Formeln} \\ n+1 & \text{Tschebyscheff-Formeln} \\ 2n+1 & \text{Gauß -Formeln} \end{cases}$$

  Dabei kann eintreten, daß der Hauptterm des Fehlers R_{n+1} nicht bei $\nu = K+1$ entsteht, sondern für größeres ν. Damit ergibt sich ein kleinerer Fehler.
  Insbesondere wird diesbezüglich bei den ersten beiden Formelgruppen eine Fallunterscheidung zwischen n gerade und ungerade durchzuführen sein.

3 wichtige Formelgruppen

Formelgruppe	Knoten und Gewichte, Anzahl der Unbekannten	Bemerkung
Newton-Cotes-Formeln	x_k äquidistant gegeben A_k gesucht $n+1,\ n-1$ bzw. n	Interpolations-Quadratur, abgeschlossene, offene, halboffene Formeln
Tschebyscheff-Formeln	$A_k = A = const$ und x_k gesucht $n+2$	x_k aus symmetrischen Funktionen und Wurzeln einer algebraischen Gleichung, $A = 1/(n+1)$ aus der Konsistenzbedingung
Gauß - Formeln	x_k und A_k gesucht $2n+2$	Gauß -Legendre-Formeln, x_k aus NS der Legendre-Polynome, A_k aus (G)

Aus der Vorgehensweise für das Aufstellen der Bestimmungsgleichungen ergeben sich 2 weitere Methoden zur Berechnung von Knoten und/oder Gewichten, die natürlich mit der bisher genannten verwandt sind.
Dabei bilden die algebraischen Polynome, die monomiale Basis $1, x, x^2, ..., x^k, ...$ sowie die Interpolation die Grundlage.

- **Interpolatorische Quadratur** (Interpolations-Quadratur)

  Bei gegebenen Stützstellen x_k vereinfacht sich die Berechnung der Gewichte unter Anwendung der Lagrange-Interpolationsformel für die Referenz $\{(x_k, f_k),\ k = 0, 1, ..., n,\ \ f_k = f(x_k)\}$ und Integration des Lagrange-Interpolationspolynoms.
  Gleichzeitig ist eine Fehlerbetrachtung auf der Grundlage des Interpolationsfehlers möglich.

  Knotenpunktpolynom $$\Phi_n(x) = \prod_{k=0}^{n} (x - x_k)$$

  Lagrange-Basispolynome $$\varphi_k(x) = \frac{\Phi_n(x)}{(x - x_k)\Phi_n'(x_k)} = \prod_{\substack{j=0 \\ j \neq k}}^{n} \frac{x - x_j}{x_k - x_j}$$

  Lagrange-Interpolationsformel $$L_n(x) = \sum_{k=0}^{n} f(x_k)\ \varphi_k(x)$$

  Interpolationsfehler $$f(x) - L_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\ \Phi_n(x),\quad \xi \in int(x_0, x_1, ..., x_n)$$

- **Genaue Integration von Polynomen maximalen Grades**

  Das bedeutet, die Bestimmungsgleichungen zu erhalten aus der Forderung, daß die Integration exakt sein soll für eine entsprechende Anzahl von Basispolynomen aus $1, x, x^2, ..., x^k, ...$

2.4 Genauigkeit der Integration und Fehlerabschätzungen

Die algebraischen Polynome

$$p_n(x) \;=\; \sum_{k=0}^{n} a_k x^{n-k} \in \mathcal{P}_n$$

spielen bei der Betrachtung des Integrationsfehlers eine zentrale Rolle.
Sei die Integrationsformel I_{n+1} auf dem linearen Raum (Klasse) $\mathcal{P}_n$ exakt, d.h.

$$I_{n+1}(p_n) = \int_a^b p_n(x)\,dx = I(p_n), \quad \forall p_n \in \mathcal{P}_n.$$

Damit beanwortet man die Frage, ob man für hinreichend großes n den Wert $I(f)$ für beliebiges $f \in C[a,b]$ beliebig genau approximieren kann.

Bemerkungen

(1) Findet man also eine Integrationsformel $I_{n+1}(f)$, die für alle Polynome vom Grade $\leq n$ exakt ist, so kann man aufgrund des Weierstraßschen Approximationssatzes (*Stetige Funktionen lassen sich beliebig genau durch Polynome approximieren.*) erwarten, daß $I_{n+1}(f)$ auch für eine umfassende Klasse von stetigen Funktionen gute Approximationen liefert.

(2) Die Integrationsformel kann aber auch beliebig schlecht sein.
Sei $I_{n+1}(f)$ eine Integrationsformel mit den Knoten x_k und Gewichten A_k und
$I(f) \approx I_{n+1}(f)$.
Für die neue Funktion

$$f_1(x) = f(x) + g(x) \prod_{k=0}^{n} (x - x_k)$$

gilt $f_1(x_k) = f(x_k)$ und damit $I_{n+1}(f_1) = I_{n+1}(f)$.
Aber durch die freie Wahl von $g(x)$ ist i.a. $I(f_1) \neq I(f)$ und der Unterschied kann beliebig groß gemacht werden.
Ohne zusätzliche Voraussetzungen über $f(x)$ kann über den Fehler $R_{n+1}(f)$ wenig ausgesagt werden.

(3) Da $I(f)$ und somit $R_{n+1}(f)$ nicht vorliegen, können Rechnungen durchgeführt und kontrolliert werden, indem i.a. 2 Näherungen $I_{n_1+1}(f)$ und $I_{n_2+1}(f)$ für Fehlerabschätzungen und Beendigung des Verfahrens herangezogen werden. Die Gültigkeit solcher Stoppregeln findet man z.B. bei der Extrapolation von Romberg (Romberg-Verfahren) für zusammengesetzte Newton-Cotes-Formeln.

2.5 Die MATLAB Funktion int

MATLAB stellt für die Berechnung von unbestimmten oder allgemeinen Integralen, also Stammfunktionen, und bestimmten Intergralen in numerischen oder symbolischen Intervallen das auf Maple basierende m-File `int` zur Verfügung. Mögliche Aufrufe mit der symbolischen Funktion $f(x)$ sind

```
r = int(f)
r = int(f,'x')
r = int(f,a,b)
r = int(f,'x',a,b)
r = int(f,'x','u','o')
```

Ergebnisse sind entweder die Stammfunktion mit der Intergrationskonstanten 0, numerische Werte (Flächeninhalte) oder symbolische Ausdrücke für das bestimmte Integral. Natürlich können auch andere Bezeichner für die Größen gewählt werden.

Das m-File nutzt die symbolischen Umformungen von Maple.

```
function r = int(f,x,a,b)
%INT Integrate.
% INT(S) is the indefinite integral of S with respect to its symbolic
% variable.  See SYMVAR for definition of "symbolic variable".
% INT(S,'v') is the indefinite integral of S with respect to v.
% INT, with no arguments, is the indefinite integral of the previous
% expression with repect to its symbolic variable.
% INT(S,a,b) is the definite integral of S with respect to its
%     symbolic variable from a to b.
% INT(S,'v',a,b) is the definite integral of S with respect to v
%     from a to b.
%
% Example: int('1/(1+x^2)') is arctan(x) .
%
% See also SYMVAR, DIFF, SYMSUM.

% Copyright (c) 1993-95 by The MathWorks, Inc.
% $Revision: 1.10 $  $Date: 1995/03/01 21:38:50 $

if nargin == 0
   f = maple('"');
elseif strcmp(f,'ans')
   f = maple('"');
end

if f(1) ~= '['
   % Scalar symbolic expression
   if nargin <= 2
```

```
      % Indefinite integral
      if nargin < 2, x = symvar(f); end
      r = maple('int',f,x);
   else
      % Definite integral
      if nargin < 4, b = a; a = x; x = symvar(f); end
      r = maple('int',f,[x '=' symrat(a) '..' symrat(b)]);
   end
else
   % Symbolic matrix.
   if nargin <= 2
      % Indefinite integral
      if nargin < 2, x = symvar(f); end
      r = maple('map','int',f,x);
   else
      % Definite integral
      if nargin < 4, b = a; a = x; x = symvar(f); end
      r = maple('map','int',f,[x '=' symrat(a) '..' symrat(b)]);
   end
end
```

Beispiel

Für die Referenz

$$\{(x_k, y_k) = (k\pi/n, f(x_k)),\ k = 0, 1, ..., n = 5,\quad f(x) = \sin x\},$$

haben wir in [11] die natürliche kubische Splinefunktion $S(x)$ unter Verwendung der Funktion `spline_k` bestimmt.

Nun soll das bestimmte Integral $\int_0^{\pi} f(x)\,dx$ und sein Näherungswert

$$\int_0^{\pi} S(x)\,dx \;=\; \sum_{k=0}^{n-1} \int_{x_k}^{x_{k+1}} s^{(k)}(x)\,dx$$

bestimmt werden.

Auf dem Intervall $[x_k, x_{k+1}]$ haben wir die Koeffizienten des natürlichen kubischen Splines

$$s^{(k)}(x) = d_k(x - x_k)^3 + c_k(x - x_k)^2 + b_k(x - x_k) + a_k,\quad k = 0, 1, ..., n - 1.$$

```
Natuerliche kubische Splines : spline_k

xi =
         0    0.6283    1.2566    1.8850    2.5133    3.1416
```

```
[x[k],x[k+1]]      d[k]*(x-x[k])^3+c[k]*(x-x[k])^2+b[k]*(x-x[k])+a[k]
                   d[k]      c[k]      b[k]      a[k]
[0.0000, 0.6283]   -0.1611    0.0000    0.9991    0.0000
[0.6283, 1.2566]   -0.0996   -0.3037    0.8083    0.5878
[1.2566, 1.8850]    0.0000   -0.4914    0.3087    0.9511
[1.8850, 2.5133]    0.0996   -0.4914   -0.3087    0.9511
[2.5133, 3.1416]    0.1611   -0.3037   -0.8083    0.5878

Sk =
d(k)*(x-x(k))^3+c(k)*(x-x(k))^2+b(k)*(x-x(k))+a(k)
```

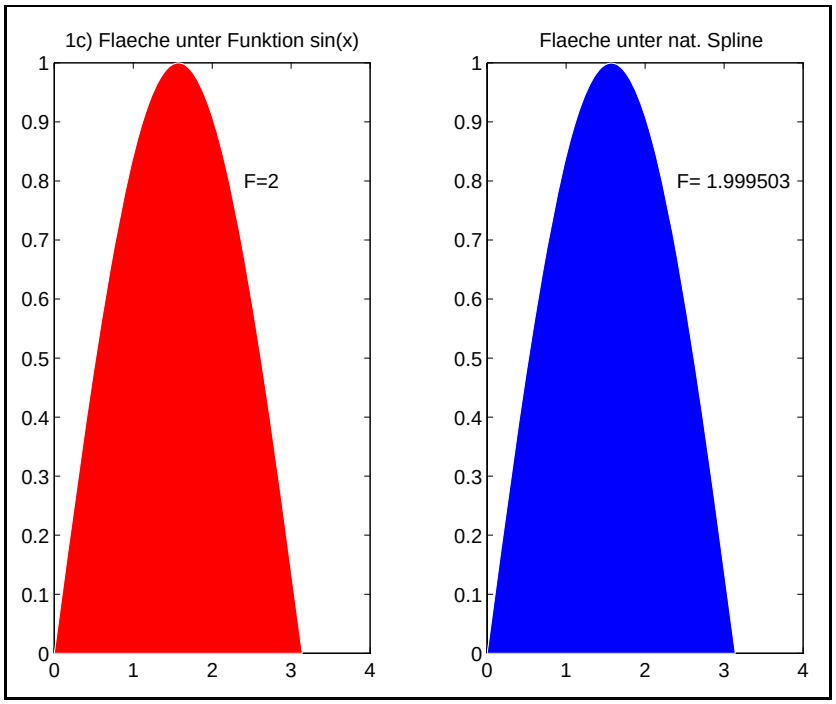
Nun führen wir die Integration aus. Die entsprechenden Vektoren $xi(1:6)$, $d, c, b, a(1:5)$ werden als gegeben vorausgesetzt.

```
n = 6;
f = 'sin(x)';
int(f)
ExakterIntegralwert = int(f,'x',0,pi)

clear k
Sk = 'd(k)*(x-xi(k))^3+c(k)*(x-xi(k))^2+b(k)*(x-xi(k))+a(k)'
int(Sk,'x')

% Summation von Teilintegralen bei Splinefunktion
summe = 0;
for k=1:n-1
 SH = expand(subs(subs(subs(subs(subs(Sk,xi(k),'xi(k)'),...
        d(k),'d(k)'),c(k),'c(k)'),...
        b(k),'b(k)'),a(k),'a(k)'));
 si = int(SH,'x',xi(k),xi(k+1));
 summe = summe+numeric(si);
end;
disp(['Naeherungswert aus Integral ueber Spline = ',sprintf('%9.6f',summe)])

% Graphik zu Flaechen
subplot(1,2,1)
xxi = linspace(0,pi,100);
yyi = sin(xxi);
fill(xxi,yyi,'r')
title('1c) Flaeche unter Funktion sin(x)')
text(2.4,0.8,'F=2')
subplot(1,2,2)
% Sm(1:5,1:4) symbolische Normalformen des Splines
xp = [];
yp = [];
```

```
for k=1:n-1
  xh = linspace(xi(k),xi(k+1),20);
  yh = polyval(Sm(k,1:4),xh);
  xp = [xp xh];
  yp = [yp yh];
end;
fill(xp,yp,'b')
title('  Flaeche unter nat. Spline')
text(2.4,0.8,['F=',sprintf('%9.6f',summe)])
print ser6gr03.ps -dps
```

Ergebnisse

```
ans =
  -cos(x)

ExakterIntegralwert =
  2

Sk =
  d(k)*(x-xi(k))^3+c(k)*(x-xi(k))^2+b(k)*(x-xi(k))+a(k)

ans =
  1/4*d(4)*(x-xi(4))^4+1/3*c(4)*(x-xi(4))^3+b(4)*(1/2*x^2-xi(4)*x)+a(4)*x

Naeherungswert aus Integral ueber Spline =  1.999503
```



2.6 Newton-Cotes-Formeln

Um einen besseren Vergleich von abgeschlossenen und offenen Formeln zu ermöglichen, wird auch der Zugang über den Abstand $\bar{h}$ der Integrationsknoten durchgeführt.
Die Stützstellen in $[-\frac{h}{2}, \frac{h}{2}]$ sind $x_k = -\frac{h}{2} + k\frac{h}{n},\ k = 0(1)n,\ \bar{h} = \frac{h}{n}$.

2.6.1 Abgeschlossene Newton-Cotes-Formeln

Integrationsknoten in $[-\frac{h}{2}, \frac{h}{2}]$

$$x_k = -\frac{h}{2} + k\frac{h}{n}, \quad k = 0(1)n$$

Integrationsgewichte A_k

$$\sum_{k=0}^{n} A_k x_k^{\nu} = \frac{1}{\nu+1}\left(\frac{h}{2}\right)^{\nu} \frac{1+(-1)^{\nu}}{2}, \quad \nu = 0(1)n$$

bzw. in ausgeschriebener Form (G1)

$$\begin{array}{lcl}
A_0 + A_1 + \cdots + A_n & = & 1 \\
A_0x_0 + A_1x_1 + \cdots + A_nx_n & = & 0 \\
A_0x_0^2 + A_1x_1^2 + \cdots + A_nx_n^2 & = & \frac{1}{3}(\frac{h}{2})^2 \\
A_0x_0^3 + A_1x_1^3 + \cdots + A_nx_n^3 & = & 0 \\
A_0x_0^4 + A_1x_1^4 + \cdots + A_nx_n^4 & = & \frac{1}{5}(\frac{h}{2})^4 \\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots & & \\
A_0x_0^n + A_1x_1^n + \cdots + A_nx_n^n & = & \frac{1}{n+1}\left(\frac{h}{2}\right)^n \frac{1+(-1)^n}{2}
\end{array}$$

Existenz und Eindeutigkeit

Zu jedem $n \in \mathbb{N}$ sind die Integrationsgewichte A_k eindeutig bestimmt.
Also existiert zu jedem $n \in \mathbb{N}$ genau eine abgeschlossene Newton-Cotes-Formel. Dies folgt aus der eindeutigen Lösbarkeit des LGS (G1) mit seiner regulären Vandermondeschen Koeffizientenmatrix.

Bemerkungen

(1) Dasselbe LGS erhält man aus der Forderung, daß die Integration exakt sein soll für die $n+1$ Basispolynome $1, x, x^2, ..., x^n$.
Damit ergibt sich die ν-te Gleichung zu

$$h\sum_{k=0}^{n} A_k x_k^{\nu} = \int_{-\frac{h}{2}}^{\frac{h}{2}} x^{\nu}\,dx = \frac{h}{\nu+1}\left(\frac{h}{2}\right)^{\nu}\frac{1+(-1)^{\nu}}{2} = \begin{cases} 0, & \text{falls } \nu \text{ ungerade} \\ h\frac{1}{\nu+1}\left(\frac{h}{2}\right)^{\nu}, & \text{falls } \nu \text{ gerade} \end{cases}$$

(2) Aufgrund der symmetrischen Lage der Knoten zum Punkt 0, erfüllen die Gewichte die Bedingung $A_k = A_{n-k}$.

(3) Mit den Symmetriebedingungen $x_{n-k} = -x_k$ und $A_k = A_{n-k}$ zeigt sich, daß eine weitere Bestimmungsgleichung aus (G1) automatisch erfüllt werden kann.
Ist nämlich n gerade, d.h. $n = 2m,\ x_m = 0$, dann lautet die $(n+2)$-te Gleichung

$$\begin{aligned} & A_0 x_0^{n+1} + A_1 x_1^{n+1} + \cdots + A_n x_n^{n+1} \\ = \ & A_0(x_0^{n+1} + x_n^{n+1}) + A_1(x_1^{n+1} + x_{n-1}^{n+1}) + \cdots + A_{m-1}(x_{m-1}^{n+1} + x_{m+1}^{n+1}) + A_m x_m^{n+1} \\ = \ & 0 \\ = \ & \frac{1}{n+2}\left(\frac{h}{2}\right)^{n+1} \frac{1+(-1)^{n+1}}{2} \end{aligned}$$

Damit erhalten wir die Exaktheit der Integrationsformel noch für das nächste Basispolynom x^{n+1}, so daß der eigentliche und hauptsächliche Fehler erst durch die nächste Differenz

$$\frac{1}{n+3}\left(\frac{h}{2}\right)^{n+2} - (A_0 x_0^{n+2} + A_1 x_1^{n+2} + \cdots + A_n x_n^{n+2}) = \mathcal{O}(h^{n+2})$$

erzeugt wird.

(4) Für n ungerade lautet die $(n+1)$-te Gleichung, die gerade noch erfüllt wird,

$$A_0 x_0^n + A_1 x_1^n + \cdots + A_n x_n^n = 0.$$

Die nächste Gleichung stellt schon den Hauptterm des Fehlers dar, also

$$\frac{1}{n+2}\left(\frac{h}{2}\right)^{n+1} - (A_0 x_0^{n+1} + A_1 x_1^{n+1} + \cdots + A_n x_n^{n+1}) = \mathcal{O}(h^{n+1}).$$

(5) Abschätzung des Integrationsfehlers auf $[-\frac{h}{2}, \frac{h}{2}]$ mittels Interpolationsquadratur und Fehler der Langrange-Interpolation.

$$\begin{aligned} \widetilde{R}_{n+1} &= \int_{-\frac{h}{2}}^{\frac{h}{2}} \frac{f^{(n+1)}(\xi)}{(n+1)!}\, \Phi_n(x)\, dx \\ |\widetilde{R}_{n+1}| &\leq \frac{\max\limits_{x\in[-h/2,h/2]} |f^{(n+1)}(x)|}{(n+1)!} \int_{-\frac{h}{2}}^{\frac{h}{2}} |\Phi_n(x)|\, dx. \end{aligned}$$

Die letzte Abschätzung des Betrages ist i.a. recht grob, weil das Knotenpunktpolynom $\Phi_n(x)$ für größeres n in $[-\frac{h}{2}, \frac{h}{2}]$ ständig das Vorzeichen wechselt.
Beachtet man Mittelwertsätze der Integralrechnung und die Vorzeicheneigenschaft von $\Phi_n(x)$ in den Teilintervallen von $[-\frac{h}{2}, \frac{h}{2}]$, so lassen sich für die einfachen NC-Formeln genauere Fehlerschätzungen herleiten.

(6) Zur Konvergenz der NC-Formeln.
Die Newton-Cotes-Formeln konvergieren nicht für jede stetige Funktion, d.h.
$\lim_{n\to\infty} I_{n+1}(f) = I(f)$ gilt nicht für beliebiges stetiges $f(x)$ (Satz von KUSMIN).
Für $n = 8$ und $n \geq 10$ treten in den NC-Formeln auch negative Gewichte A_k auf. Durch Subtraktionen können dann große Rundungsfehler entstehen.
Für $n \to \infty$ wächst $\max_{k=0(1)n} |A_k|$ unbegrenzt, was ebenfalls für numerische Rechnungen ungünstig ist.

2.6.2 Die einfachsten halbabgeschlossenen NC-Formeln

Damit sind die Rechteckregeln gemeint, die auch als halboffene NC-Formeln bezeichnet werden.

Rechteckregel (Links) $(n = 0)$

Knoten	:	$x_0 = -\frac{h}{2}$
Gewichte	:	$A_0 = 1$ (eine Bestimmungsgleichung)
Formel	:	$R_L = h\ f(-\frac{h}{2})$
Fehler	:	$F_{R_L} = I - R_L = \frac{h^2}{2} f'(\xi), \quad -\frac{h}{2} \leq \xi \leq \frac{h}{2}$

Berechnung des Fehlergliedes gemäß Bestimmungsgleichungen (G), $K = 1$

$$
\begin{aligned}
R_1 &= h \sum_{\nu=1}^{\infty} \left\{ \frac{1}{\nu+1} \left(\frac{h}{2}\right)^{\nu} \frac{1+(-1)^{\nu}}{2} - A_0 x_0^{\nu} \right\} a_{\nu}, \quad a_{\nu} = \frac{f^{(\nu)}(0)}{\nu!} \\
&= h \sum_{\nu=1}^{1} \left\{ \frac{1}{\nu+1} \left(\frac{h}{2}\right)^{\nu} \frac{1+(-1)^{\nu}}{2} - A_0 x_0^{\nu} \right\} a_1, \quad a_1 = \frac{f'(\xi)}{1!} \\
&= h \left(0 - \left(-\frac{h}{2}\right)^1\right) f'(\xi) \\
&= \frac{h^2}{2}\ f'(\xi), \quad \xi \in [-h/2, h/2]
\end{aligned}
$$

Analoges Ergebnis erhält man mit dem Interpolationsquadraturfehler.

Rechteckregel (Rechts) $(n = 0)$

Knoten	:	$x_0 = +\frac{h}{2}$
Gewichte	:	$A_0 = 1$
Formel	:	$R_R = h\ f(\frac{h}{2})$
Fehler	:	$F_{R_R} = I - R_R = -\frac{h^2}{2} f'(\xi), \quad -\frac{h}{2} \leq \xi \leq \frac{h}{2}$

2.6.3 Die gebräuchlichsten abgeschlossenen NC-Formeln

Zunächst betrachten wir die Trapezregel (Sehnentrapezregel) und die Simpsonregel (Keplerregel).

Trapezregel $(n = 1)$

Knoten : $x_0 = -\frac{h}{2}, \quad x_1 = \frac{h}{2}$

(G) :
$$\begin{aligned} A_0 \; + \; A_1 \; &= 1 \\ -\tfrac{h}{2}A_0 \; + \; \tfrac{h}{2}A_1 \; &= 0 \\ \tfrac{h^2}{4}A_0 \; + \; \tfrac{h^2}{4}A_1 \; &\neq \tfrac{1}{3}\left(\tfrac{h}{2}\right)^2 \quad \text{(nicht erfüllt)} \end{aligned}$$

Gewichte : $A_0 = \frac{1}{2}, \quad A_1 = \frac{1}{2}$

$$A_0 = \tfrac{1}{h}w_0 = \tfrac{1}{h}I(\varphi_0) = \int\limits_{-\frac{h}{2}}^{\frac{h}{2}} \tfrac{x-h/2}{-h}\, dx$$

$$A_1 = \tfrac{1}{h}w_1 = \tfrac{1}{h}I(\varphi_1) = \int\limits_{-\frac{h}{2}}^{\frac{h}{2}} \tfrac{x+h/2}{h}\, dx$$

Formel : $T = \dfrac{h}{2}\left[f\Big(-\dfrac{h}{2}\Big) + f\Big(\dfrac{h}{2}\Big)\right]$

Fehler : $F_T = I - T = -\dfrac{h^3}{6}\dfrac{f''(\xi)}{2!}, \quad -\dfrac{h}{2} \le \xi \le \dfrac{h}{2}$

Berechnung des Fehlergliedes gemäß Interpolationsquadraturfehler

$$\begin{aligned} \widetilde{R}_2 &= \int\limits_{-\frac{h}{2}}^{\frac{h}{2}} \frac{f^{(2)}(\xi)}{2!}\, \Phi_1(x)\, dx, \quad \Phi_1(x) = (x-x_0)(x-x_1) = (x+h/2)(x-h/2) \\ &\quad \Phi_1(x) \text{ hat keinen Vorzeichenwechsel in } [-h/2, h/2], \\ &= \frac{1}{2}f''(\bar{\xi}) \int\limits_{-\frac{h}{2}}^{\frac{h}{2}} (x^2 - (h/2)^2)\, dx \\ &= \frac{1}{2}f''(\bar{\xi}) \left[\frac{x^3}{3} - \Big(\frac{h}{2}\Big)^2 x\right]_{-h/2}^{h/2} \\ &= -\frac{h^3}{12}\, f''(\bar{\xi}) \end{aligned}$$

Simpsonregel $(n = 2)$

Knoten : $x_0 = -\frac{h}{2}, \quad x_1 = 0, \quad x_2 = \frac{h}{2}$

(G) :
$$\begin{aligned} A_0 \;+\; A_1 \;+\; A_2 &= 1 \\ -\tfrac{h}{2}A_0 \;+\; 0\cdot A_1 \;+\; \tfrac{h}{2}A_2 &= 0 \\ \tfrac{h^2}{4}A_0 \;+\; 0\cdot A_1 \;+\; \tfrac{h^2}{4}A_2 &= \tfrac{1}{3}\left(\tfrac{h}{2}\right)^2 \\ -\tfrac{h^3}{8}A_0 \;+\; 0\cdot A_1 \;+\; \tfrac{h^3}{8}A_2 &= 0 \quad \text{(zusätzlich erfüllt)} \\ \tfrac{h^4}{16}A_0 \;+\; 0\cdot A_1 \;+\; \tfrac{h^4}{16}A_2 &\neq \tfrac{1}{5}\left(\tfrac{h}{2}\right)^4 \quad \text{(nicht erfüllt)} \end{aligned}$$

Gewichte : $A_0 = A_2 = \frac{1}{6}, \quad A_1 = \frac{4}{6}$

$$A_0 = \tfrac{1}{h}w_0 = \tfrac{1}{h}I(\varphi_0) = \int\limits_{-\frac{h}{2}}^{\frac{h}{2}} \tfrac{x(x-h/2)}{h^2/2}\,dx$$

$$A_1 = \tfrac{1}{h}w_1 = \tfrac{1}{h}I(\varphi_1) = \int\limits_{-\frac{h}{2}}^{\frac{h}{2}} \tfrac{x^2-(h/2)^2}{-h^2/4}\,dx$$

$$A_2 = \tfrac{1}{h}w_2 = \tfrac{1}{h}I(\varphi_2) = \int\limits_{-\frac{h}{2}}^{\frac{h}{2}} \tfrac{(x+h/2)x}{h^2/2}\,dx$$

Formel : $$S = \frac{h}{6}\left[f\Big(-\frac{h}{2}\Big) + 4f(0) + f\Big(\frac{h}{2}\Big)\right]$$

Fehler : $$F_S = I - S = -\frac{h^5}{120}\frac{f^{(4)}(\xi)}{4!}, \quad -\frac{h}{2} \le \xi \le \frac{h}{2}$$

Berechnung des Fehlergliedes gemäß Bestimmungsgleichungen (G), $K = 3 \Rightarrow 4$

$$\begin{aligned} R_3 &= h\sum_{\nu=4}^{\infty}\left\{\frac{1}{\nu+1}\left(\frac{h}{2}\right)^{\nu}\frac{1+(-1)^{\nu}}{2} - (A_0x_0^{\nu} + A_1x_1^{\nu} + A_2x_2^{\nu})\right\}a_{\nu}, \quad a_{\nu} = \frac{f^{(\nu)}(0)}{\nu!} \\ &= h\sum_{\nu=4}^{4}\left\{\frac{1}{\nu+1}\left(\frac{h}{2}\right)^{\nu}\frac{1+(-1)^{\nu}}{2} - (A_0x_0^{\nu} + A_1x_1^{\nu} + A_2x_2^{\nu})\right\}a_4, \quad a_4 = \frac{f^{(4)}(\xi)}{4!} \\ &= h\left\{\frac{1}{5}\Big(\frac{h}{2}\Big)^4 - \Big[\frac{1}{6}\Big(-\frac{h}{2}\Big)^4 + \frac{4}{6}0^4 + \frac{1}{6}\Big(\frac{h}{2}\Big)^4\Big]\right\}\frac{f^{(4)}(\xi)}{24} \\ &= -\frac{h^5}{2880}\,f^{(4)}(\xi), \quad \xi \in [-h/2, h/2] \end{aligned}$$

Man beachte, daß im Quadraturfehler $\mathcal{O}(h^p)$ eine h-Potenz durch die Standard-Intervallänge h zustande kommt.
Die Integration über ein Intervall $[a, b]$ wird die Fehlerordnung $\mathcal{O}(h^{p-1})$ haben, wenn h die Länge des Teilintervalls ist.

2.6.4 Übersichten

n	Bezeichnung	Formel I_{n+1}	Fehler R_{n+1} $\bar{h}=\frac{h}{n},\ \xi\in[-\frac{h}{2},\frac{h}{2}]$
1	N_2, T Trapezregel Sehnentrapezr.	$\frac{h}{2}[f(-\frac{h}{2})+f(\frac{h}{2})]$	$-\frac{h^3}{12}f''(\xi)$ $=-\frac{h^3}{6}\frac{f''(\xi)}{2!}$ $=-\frac{1}{12}\bar{h}^3 f''(\xi)$
2	N_3, S Simpsonregel Keplerregel	$\frac{h}{6}[f(\frac{-h}{2})+4f(0)+f(\frac{h}{2})]$	$-\frac{h^5}{2880}f^{(4)}(\xi)$ $=-\frac{h^5}{120}\frac{f^{(4)}(\xi)}{4!}$ $=-\frac{1}{90}\bar{h}^5 f^{(4)}(\xi)$
3	N_4, N-3/8 Newton-3/8-R. 3/8-Regel	$\frac{h}{8}[f(\frac{-h}{2})+3f(\frac{-h}{6})+3f(\frac{h}{6})+f(\frac{h}{2})]$	$-\frac{h^5}{6480}f^{(4)}(\xi)$ $=-\frac{h^5}{270}\frac{f^{(4)}(\xi)}{4!}$ $=-\frac{3}{80}\bar{h}^5 f^{(4)}(\xi)$
4	N_5, M Milneregel 1/90-Regel	$\frac{h}{90}[7f(\frac{-h}{2})+32f(\frac{-h}{4})+12f(0)$ $+32f(\frac{h}{4})+7f(\frac{h}{2})]$	$-\frac{h^7}{1\,935\,360}f^{(6)}(\xi)$ $=-\frac{h^7}{2688}\frac{f^{(6)}(\xi)}{6!}$ $=-\frac{8}{945}\bar{h}^7 f^{(6)}(\xi)$
5	N_6 1/288-Regel	$\frac{h}{288}[19f(\frac{-h}{2})+75f(\frac{-3h}{10})+50f(\frac{-h}{10})$ $+50f(\frac{h}{10})+75f(\frac{3h}{10})+19f(\frac{h}{2})]$	$-\frac{11\,h^7}{37\,800\,000}f^{(6)}(\xi)=$ $-\frac{11\,h^7}{52500}\frac{f^{(6)}(\xi)}{6!}$ $-\frac{275}{12096}\bar{h}^7 f^{(6)}(\xi)$
6	N_7 1/840-Regel	$\frac{h}{840}[41f(\frac{-h}{2})+216f(\frac{-h}{3})+27f(\frac{-h}{6})$ $+272f(0)+27f(\frac{h}{6})+216f(\frac{h}{3})]$ $+41f(\frac{h}{2})]$	$=-\frac{h^9}{1\,567\,641\,600}f^{(8)}(\xi)$ $=-\frac{h^9}{38880}\frac{f^{(8)}(\xi)}{8!}$ $-\frac{9}{1400}\bar{h}^9 f^{(8)}(\xi)$
7	N_8 1/17280-Regel	$\frac{h}{17280}[751f(\frac{-h}{2})+3577f(\frac{-5h}{14})+1323f(\frac{-3h}{14})$ $+2989f(\frac{-h}{14})+2989f(\frac{h}{14})+1323f(\frac{3h}{14})$ $+3577f(\frac{5h}{14})+751f(\frac{h}{2})]$	$-\frac{167\,h^9}{426\,924\,691\,200}f^{(8)}(\xi)$ $=-\frac{167\,h^9}{10\,588\,410}\frac{f^{(8)}(\xi)}{8!}$ $=-\frac{8183}{518\,400}\bar{h}^9 f^{(8)}(\xi)$
8	N_9 1/28350-Regel	$\frac{h}{28350}[989f(\frac{-h}{2})+5888f(\frac{-3h}{8})-928f(\frac{-h}{4})$ $+10946f(\frac{-h}{8})-4540f(0)+10946f(\frac{h}{8})$ $-928f(\frac{h}{4})+5888f(\frac{3h}{8})+989f(\frac{h}{2})]$	$-\frac{37\,h^{11}}{62\,783\,697\,715\,200}f^{(10)}(\xi)$ $=-\frac{37\,h^{11}}{17\,301\,504}\frac{f^{(10)}(\xi)}{10!}$ $=-\frac{2368}{467\,775}\bar{h}^{11}f^{(10)}(\xi)$

Übersichtlicher ist eine Zusammenstellung der Gewichte und Koeffizienten gemäß folgender Tabelle auf der Basis der Formeln

$$
\begin{aligned}
I_{n+1} &= \frac{h}{A}\sum_{k=0}^{n} W_k f(x_k) \\
R_{n+1} &= \begin{cases} -Ch^{n+2}f^{(n+1)}(\xi), & \text{falls } n \text{ ungerade} \\ -Ch^{n+3}f^{(n+2)}(\xi), & \text{falls } n \text{ gerade} \end{cases} \\
&= \begin{cases} -D\bar{h}^{n+2}f^{(n+1)}(\xi), & \text{falls } n \text{ ungerade} \\ -D\bar{h}^{n+3}f^{(n+2)}(\xi), & \text{falls } n \text{ gerade} \end{cases}
\end{aligned}
$$

Aus Symmetriegründen braucht man nur die Gewichte $W_0, W_1, ..., W_{[\frac{n}{2}]}$ zu notieren.

n	A	W_0	W_1	W_2	W_3	W_4	W_5	C	D
1	2	1						$\frac{1}{12}$	$\frac{1}{12}$
2	6	1	4					$\frac{1}{2880}$	$\frac{1}{90}$
3	8	1	3					$\frac{1}{6480}$	$\frac{3}{80}$
4	90	7	32	12				$\frac{1}{1\,935\,360}$	$\frac{8}{945}$
5	288	19	75	50				$\frac{11}{37\,800\,000}$	$\frac{275}{12096}$
6	840	41	216	27	272			$\frac{1}{1\,567\,641\,000}$	$\frac{9}{1400}$
7	17280	751	3577	1323	2989			$\frac{167}{426\,924\,691\,200}$	$\frac{8183}{518\,400}$
8	28350	989	5888	-928	10946	-4540		$\frac{37}{62\,783\,697\,715\,200}$	$\frac{2368}{467\,775}$
9	89600	2857	15741	1080	19344	5778		$\frac{173}{458\,209\,960\,750\,080}$	$\frac{4671}{394\,240}$
10	598752	16067	106300	-48525	272400	-260550	427368	$\frac{26927}{653\,837\,184\cdot 10^{11}}$	$\frac{673\,175}{163\,459\,296}$

2.6.5 Offene Newton-Cotes-Formeln

Integrationsknoten in $[-\frac{h}{2}, \frac{h}{2}]$

$$x_k = -\frac{h}{2} + k\frac{h}{n}, \quad k = 1(1)n-1$$

Integrationsgewichte A_k

$$\sum_{k=1}^{n-1} A_k x_k^{\nu} = \frac{1}{\nu+1}\left(\frac{h}{2}\right)^{\nu}\frac{1+(-1)^{\nu}}{2}, \quad \nu = 0(1)n-2$$

bzw. in ausgeschriebener Form (G2)

$$\begin{array}{lllll}
A_1 & + A_2 & + \cdots + A_{n-1} & = & 1 \\
A_1 x_1 & + A_2 x_2 & + \cdots + A_{n-1} x_{n-1} & = & 0 \\
A_1 x_1^2 & + A_2 x_2^2 & + \cdots + A_{n-1} x_{n-1}^2 & = & \frac{1}{3}(\frac{h}{2})^2 \\
A_1 x_1^3 & + A_2 x_2^3 & + \cdots + A_{n-1} x_{n-1}^3 & = & 0 \\
A_1 x_1^4 & + A_2 x_2^4 & + \cdots + A_{n-1} x_{n-1}^4 & = & \frac{1}{5}(\frac{h}{2})^4 \\
\cdots & & & & \\
A_1 x_1^{n-2} & + A_2 x_2^{n-2} & + \cdots + A_{n-1} x_{n-1}^{n-2} & = & \frac{1}{n-1}\left(\frac{h}{2}\right)^{n-2} \frac{1+(-1)^{n-2}}{2}
\end{array}$$

Die einfachste offene Formel

Rechteckregel (Mitte), Mittelpunktregel, Tangententrapezregel $(n = 2)$

Knoten : $x_1 = 0$

(G2) : $A_1 \quad = 1$

$A_1 x_1 = 0$ (zusätzlich erfüllt)

$A_1 x_1^2 \neq \frac{1}{3}\left(\frac{h}{2}\right)^2$ (nicht erfüllt)

Gewichte : $A_1 = \frac{1}{h} w_1 = \frac{1}{h} I(\varphi_1) = \frac{1}{h} \int_{-\frac{h}{2}}^{\frac{h}{2}} 1 \, dx = 1$

Formel : $R_M = h \; f(0)$

Fehler : $F_{R_M} = I - R_M = \dfrac{h^3}{12} \dfrac{f''(\xi)}{2!}, \quad -\dfrac{h}{2} \leq \xi \leq \dfrac{h}{2}$

Für hinreichend glatten Integranden ist die Mittelpunktregel nicht nur genauer als die Rechteckregeln, sondern wegen des kleineren Faktors $\frac{1}{24}$ auch besser als die Trapezregel, trotz der halben Anzahl der benötigten Funktionswertberechnungen.
Bei zusammengesetzten Quadraturformeln ist natürlich das Verhältnis der Anzahlen der Funktionsauswertungen $anz(T) : anz(R_M) = N + 1 : N$, also kein signifikanter Unterschied dieser Anzahlen festzustellen.

Übersicht

n	Bez.	Formel I_{n-1}	Fehler R_{n-1} $\bar{h} = \frac{h}{n}, \quad \xi \in [-\frac{h}{2}, \frac{h}{2}]$
2	O_1, R_M	$hf(0)$	$\frac{h^3}{24} f''(\xi) = \frac{h^3}{12} \frac{f''(\xi)}{2!} = \frac{1}{3} \bar{h}^3 f''(\xi)$
3	O_2	$\frac{h}{2}[f(\frac{-h}{6}) + f(\frac{h}{6})]$	$\frac{h^3}{36} f''(\xi) = \frac{h^3}{18} \frac{f''(\xi)}{2!} = \frac{3}{4} \bar{h}^3 f''(\xi)$
4	O_3	$\frac{h}{3}[2f(\frac{-h}{4}) - f(0) + 2f(\frac{h}{4})]$	$\frac{7\,h^5}{23\,040} f^{(4)}(\xi)$ $= \frac{7\,h^5}{960} \frac{f^{(4)}(\xi)}{4!} = \frac{14}{45} \bar{h}^5 f^{(4)}(\xi)$
5	O_4	$\frac{h}{24}[11f(\frac{-3h}{10}) + f(\frac{-h}{10}) + f(\frac{h}{10}) + 11f(\frac{3h}{10})]$	$\frac{19\,h^5}{90\,000} f^{(4)}(\xi)$ $= \frac{19\,h^5}{3750} \frac{f^{(4)}(\xi)}{4!} = \frac{95}{144} \bar{h}^5 f^{(4)}(\xi)$

2.7 Zusammengesetzte Quadraturformeln

2.7.1 Grundlagen

- **Motivation**

  Einfache Newton-Cotes-Formeln konvergieren nicht für jede stetige Funktion, d.h. $\lim\limits_{n\to\infty} I_n = I$ gilt nicht für beliebiges stetiges $f(x)$ (Satz von KUSMIN).

  Ausweg:

  Man unterteilt das Gesamtintervall $[a,b]$ (der Einfachheit halber) in N gleichlange Teilintervalle der Schrittweite $h = (b-a)/N$ und wendet in jedem Teilintervall eine der einfachen Quadraturformeln an.

- **Problemstellung**

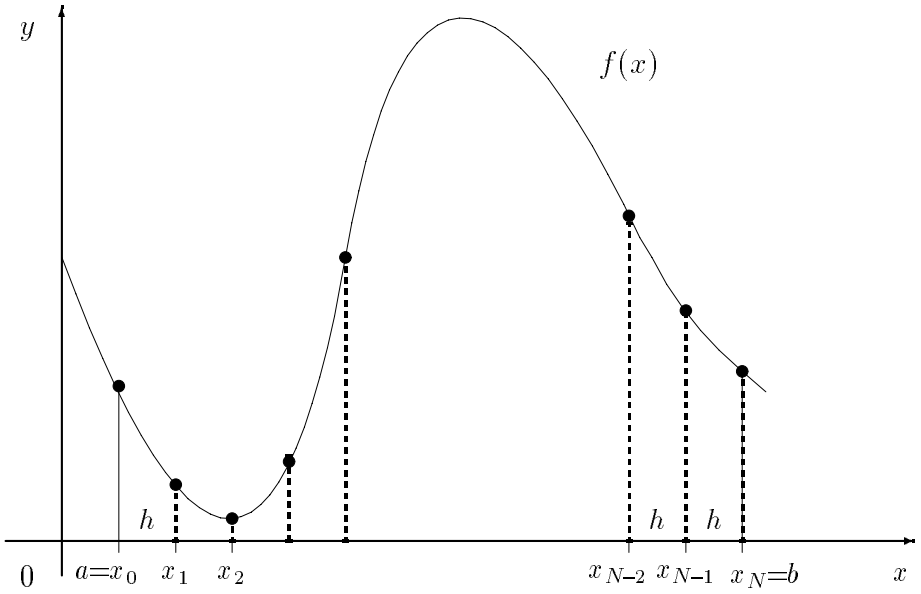
  Approximation des bestimmten Integrals

$$I = I(f) = I_a^b(f) = \int_a^b f(x)dx\ , \quad -\infty < a < b < \infty$$

- **Intervallzerlegung**

  N Teilintervalle : gleich große ($h = x_{k+1} - x_k = (b-a)/N$)
  bzw. nicht uniform ($h_k = x_{k+1} - x_k$)

  $Z : a = x_0 < x_1 < ... < x_N = b$ heißt **Zerlegung** der Integrationsintervalls.



- **Teilintegrale**

  Auf $[x_k, x_{k+1}]$ betrachtet man $I = \int_{x_k}^{x_{k+1}} f(x)\,dx$.

- **Transformation auf Standardintervall** (Referenzintervall)

$$\Big[-\frac{h}{2},\frac{h}{2}\Big] \quad \text{und} \quad I=\int\limits_{-h/2}^{h/2} f(x)dx$$

- **Zusammengesetzte Quadraturformel**

  Auf jedem Teilintervall verwendet man eine (möglichst einfache) Quadraturformel. Die Addition der Teilergebnisse liefert die *summierte* oder *zusammengesetzte Quadraturformel.*

  Eine solche Vorgehensweise ist dann angebracht, falls der Integrand nicht die entsprechende Glattheit besitzt, und man die Genauigkeit eher durch eine kleine Teilintervalllänge als durch ein hohe Ableitung im Fehlerglied erreichen will.

- **Lokaler und globaler Quadraturfehler**

  Die Differenz aus Integralwert und Quadraturformel für das Referenzintervall liefert den sogenannten *lokalen Quadraturfehler*, die Differenz aus Integralwert und zusammengesetzter Quadraturformel den *globalen Quadraturfehler.*

  Bei geeigneter Zerlegung des Intervalls kann der Fehler beliebig klein gemacht werden. Aber gleichzeitig ist natürlich auch der Rechenaufwand (Anzahl der Funktionswertberechnungen) zu beachten, so daß optimale Gewichte und geeignete Knoten schon wichtig sind.

- **Lokale und globale Fehlerordnung**

  Ist eine für ein Referenzintervall $[t_0,t_1]$ konstruierte Quadraturformel exakt für alle Polynome vom Grad $\le n$ und gilt $f\in C^{n+1}[t_0,t_1]$, so beträgt die *lokale Fehlerordnung* $p=n+2$, d.h. $\mathcal{O}((t_1-t_0)^{n+2})$.

  Diese Fehler haben wir auf dem Referenzintervall $[-\frac{h}{2},\frac{h}{2}]$ in der Form Ch^{n+2} bislang erhalten. Für besondere Stützstellenverteilungen und entsprechender Glattheit von f konnte diese Fehlerordnung in einigen Fällen sogar noch höher liegen (meist um 1).

  Ist $f\in C^{n+1}[a,b]$ und ist h_{max} die Länge des größten Teilintervalls der Zerlegung Z von $[a,b]$, so beträgt die Fehlerordnung der aus Teilen zusammengesetzten Quadraturformel $\mathcal{O}(h_{max}^{n+1})$, d.h. die globale Fehlerordnung ist $p=n+1$. Durch die Akkumulation der Fehler über N Teilintervalle ist der globale Fehler in der Ordnung um 1 niedriger.

  Der Abstand $\bar{h}$ der Stützstellen spielt in den zusammengesetzten Quadraturformeln eher eine untergeordnete Rolle.

- **Übersicht zu einfachen Formeln und ihren Fehlern**

  Man unterteilt das Gesamtintervall $[a,b]$ (der Einfachheit halber) in N Teilintervalle der Länge $h=(b-a)/N$ und wendet auf jedem **Teilintervall** eine der bisherigen einfachen Quadraturformeln an.

Abgeschlossene NC-Formeln: Stützstellen $x_0, x_1, ..., x_n$, Anzahl=$n+1$

$$I = N_{n+1} + \begin{cases} Ch^{n+2} f^{(n+1)}(\xi), & \text{falls } n \text{ ungerade} \\ Ch^{n+3} f^{(n+2)}(\xi), & \text{falls } n \text{ gerade} \end{cases} \qquad n = 1, 2, ...$$

Offene NC-Formeln: Stützstellen $x_1, x_2, ..., x_{n-1}$, Anzahl=$n-1$

$$I = O_{n-1} + \begin{cases} Ch^{n} f^{(n-1)}(\xi), & \text{falls } n \text{ ungerade} \\ Ch^{n+1} f^{(n)}(\xi), & \text{falls } n \text{ gerade} \end{cases} \qquad n = 2, 3, ...$$

- **Ausgewählte zusammengesetzte Näherungsformeln**

Rechteckr. (Links) : $R_L = h\,[f(x_0) + f(x_1) + \ldots + f(x_{N-1})],\ x_k = a + kh$
$I = R_L + (b-a)\frac{h}{2} f'(\xi), \quad a \le \xi \le b$

Rechteckr. (Rechts) : $R_R = h\,[f(x_1) + f(x_2) + \ldots + f(x_N)],\ x_k = a + kh$
$I = R_R - (b-a)\frac{h}{2} f'(\xi), \quad a \le \xi \le b$

Rechteckr. (Mitte) : $R_M = h\,[f(x_0 + \frac{h}{2}) + f(x_1 + \frac{h}{2}) + \ldots + f(x_{N-1} + \frac{h}{2})]$
$I = R_M + (b-a)\frac{h^2}{24} f''(\xi)$

Trapezregel : $T = h\,[f(x_0) + 2f(x_1) + \ldots + 2f(x_{N-1}) + f(x_N)]$
$I = T - (b-a)\frac{h^2}{12} f''(\xi)$

Simpsonregel : $S = \frac{h}{6}[f(x_0) + 4f(x_1) + 2f(x_2) + \ldots$
$+4f(x_{2N-1}) + f(x_{2N})], \quad x_k = a + k\frac{h}{2}$
$I = S - (b-a)\frac{h^4}{2880} f^{(4)}(\xi), \quad a \le \xi \le b$

Newton 3/8 Regel : $N_4 = \frac{h}{8}[f(x_0) + 3f(x_1) + 3f(x_2) + 2f(x_3) + 3f(x_4) + \ldots$
$+3f(x_{3N-1}) + f(x_{3N})], \quad x_k = a + k\frac{h}{3}$
$I = N_4 - (b-a)\frac{h^4}{6480} f^{(4)}(\xi), \quad a \le \xi \le b$

- **Fehlerbetrachtung bei R_L**

Der Gesamtfehler ergibt sich als Summe der Einzelfehler.

$$\begin{aligned} F_{R_L} &= \frac{h^2}{2} f'(\xi_0) + \frac{h^2}{2} f'(\xi_1) + ... + \frac{h^2}{2} f'(\xi_N), \quad x_k \le \xi_k \le x_{k+1} \\ &= \frac{h}{2}\, hN\ f'(\xi), \quad x_0 \le \xi \le x_N \\ &= (b-a)\frac{h}{2}\ f'(\xi) \end{aligned}$$

- **Fehlerbetrachtung bei zusammengesetzten Formeln**

  Wir verweisen auf die unterschiedlichen Darstellungen in der Literatur.
  Wichtig ist dabei zu erkennen, welche Bedeutung die Größe h hat, ob es sich dabei um die Länge des Teilintervalls oder um den Abstand der Stützstellen handelt.
  Nehmen wir die zusammengesetzte Simpsonregel.
  Meistens ist $h = \frac{b-a}{N}$ die Teilintervalllänge. Mit dem Abstand der Stützstellen $\bar{h} = \frac{h}{2}$ ergeben sich daraus die Darstellungen

  $$S = \tfrac{2\bar{h}}{6}\left[f(x_0) + 4f(x_1) + 2f(x_2) + \ldots + 4f(x_{2N-1}) + f(x_{2N})\right],\ x_k = a + k\bar{h},$$

  $$I = S - \tfrac{b-a}{2880}h^4 f^{(4)}(\xi) = S - \tfrac{b-a}{180}\bar{h}^4 f^{(4)}(\xi) = S - \left(\tfrac{b-a}{2}\right)^5 \tfrac{1}{90N^4} f^{(4)}(\xi).$$

  Natürlich wählt man hier die Anzahl der Intervalle $[x_k, x_{k+1}]$ geradzahlig, um über jeweils zwei aufeinanderfolgende mit der Simpsonregel zu integrieren.
  Wir wählen nachfolgend eine geeignete fortlaufende Indizierung der Knoten.

- **Einfache und zweifach zusammengesetzte Formeln**

  Betrachten wir auf dem Intervall $[a, b]$, $h = b - a$, die einfache Trapez- und Simpsonregel sowie die entsprechenden zusammengesetzten bei **einer** Intervallhalbierung. Es gilt folgendes.

  Trapezregel

  $$\begin{aligned} I &= T(h) - h\frac{h^2}{12}f''(\xi_1) = T([a,b]) - \frac{h^3}{12}f''(\xi_1) \\ I &= T_z(h/2) - h\frac{(h/2)^2}{12}f''(\xi_2) \\ &= T([a, a+h/2]) + T([a+h/2, b]) - \frac{1}{4}\frac{h^3}{12}f''(\xi_2) \end{aligned}$$

  Sind die Ableitungen vergleichbar, so ist auf $[a, b]$ die zweifach zusammengesetzte Trapezregel $T_z(h/2)$ ungefähr 4 Mal genauer als die einfache Trapezregel.

  Simpsonregel

  $$\begin{aligned} I &= S(h) - h\frac{h^4}{2880}f^{(4)}(\xi_1) = S([a,b]) - \frac{h^5}{2880}f^{(4)}(\xi_1) \\ I &= S_z(h/2) - h\frac{(h/2)^4}{2880}f^{(4)}(\xi_2) \\ &= S([a, a+h/2]) + S([a+h/2, b]) - \frac{1}{16}\frac{h^5}{2880}f^{(4)}(\xi_2) \end{aligned}$$

  Sind auch hier die Ableitungen vergleichbar, so ist auf $[a, b]$ die zweifach zusammengesetzte Simpsonregel $S_z(h/2)$ ungefähr 16 Mal genauer als die einfache Simpsonregel.

Bei adaptiven Methoden werden beide, also Grobrechnung (einfache Regel) und Feinrechnung (zusammengesetzte Regel) verglichen bezüglich der Genauigkeitsforderung ε, um zu entscheiden, ob die Feinrechnung als Approximation akzeptabel ist.
Wenn die Approximation nicht hinreichend genau ist, kann die Fehlerabschätzung individuell auf die beiden Teilintervalle angewandt werden, um festzustellen, ob die Approximation des Intergrals auf jedem Teilintervall innerhalb einer Toleranz $\frac{1}{2}\varepsilon$ liegt. Ist dies so, stimmt die Summe der Approximationen mit $\int_a^b f(x)dx$ innerhalb der Toleranz ε überein. Diese Intervallhalbierung kann fortgesetzt werden. Die Erhöhung der Toleranz dabei um den Faktor 2 ist vernünftig, da jede Unterteilung die Genauigkeit der Approximation um den Faktor 4 bzw. 16 erhöht.
Bei starken Schwankungen der Ableitungen des Integranden ist es jedoch angebracht, einen Sicherheitsparameter bezüglich der Toleranzverkleinerung vorzusehen. Praktisch arbeitet man mit $\rho\varepsilon,\ \rho \in [\frac{1}{2}, 1)$.

2.7.2 Zusammengesetzte Newton-Cotes-Formeln

Stützstellenbezeichnung (abweichend von der ersten Definition in der Zerlegung)

$$x_i = a + i\bar{h}, \quad i = 0(1)nN \quad \text{mit} \quad h = \frac{b-a}{N}, \quad \bar{h} = \frac{h}{n}.$$

Die Technik der Konstruktion dieser Formeln wurde bei der Simpsonregel deutlich. Bezüglich der Darstellung des Fehlers können wir ausgehend von $Ch^{n+r+1}f^{(n+r)}$, $r = 1, 2,$ das Restglied der zusammengesetzten Formel $N_{n+1},\ n = 1, 2, ...,$ nunmehr ermitteln mit $Nh = b - a$ gemäß

$$R_{n+1} = N\ Ch^{n+r+1}f^{(n+r)} = (b-a)Ch^{n+r}f^{(n+r)} = (b-a)Cn^{n+r}\bar{h}^{n+r}f^{(n+r)}.$$

Für die Newton-3/8-Regel ($n = 3$) erhalten wir

$$\begin{aligned} N_4 &= \frac{h}{8}[f(x_0) + 3f(x_1) + 3f(x_2) + 2f(x_3) + 3f(x_4) + \ldots \\ &\qquad +3f(x_{3N-2}) + 3f(x_{3N-1}) + f(x_{3N})], \quad x_i = a + i\frac{h}{3}, \\ R_4 &= I - N_4 = -\frac{b-a}{6480}h^4 f^{(4)}(\xi) = -\frac{b-a}{80}\left(\frac{h}{3}\right)^4 f^{(4)}(\xi),\ a \le \xi \le b. \end{aligned}$$

Fehlerordnung einer Integrationsformel

Die zusammengesetzte Integrationsformel N_{n+1} besitzt die *Fehlerordnung* $p \in \mathbb{N}$ (bezüglich der Schrittweite h), falls Konstanten $h_0 > 0$ und $C > 0$ existieren, so daß

$$|R_{n+1}| = |I - N_{n+1}| \le C\,h^p$$

für alle h mit $0 < h \le h_0$ gilt. Schreibweise mit *Landau-Symbol*: $R_{n+1} = \mathcal{O}(h^p)$.

Die Ordnung p ist bei NC-Formeln entweder $n+1$ oder $n+2$.

Vergleicht man für eine hinreichend glatte Funktion f die Trapezregel und Simpsonregel mit demselben Aufwand an Funktionsberechnungen (gleicher Abstand $\bar{h}$ der Stützstellen), so hat S den kleineren Fehler $-\frac{b-a}{180}\,\bar{h}^4 f^{(4)}$ gegenüber $-\frac{b-a}{12}\,\bar{h}^2 f^{(2)}$ von T und liefert somit die wesentlich besseren Näherungen.

Beispiel 1

$$I = \int_0^{\pi} \sin x \; dx = 2$$

Trapezregel $T = N_2$ $(n = 1,\ p = 2)$ und Simpsonregel $S = N_3$ $(n = 2,\ p = 4)$ liefern für I die Näherungwerte I_N mit der Zahl $K = nN + 1$ der benutzten Knoten.

Formel	N	I_N	K	$I - I_N$	q
Trapezregel	40	1.998971811	41	1.03E-3	4.0019
	80	1.999742972	81	2.57E-4	4.0000
	160	1.999935744	161	6.44E-5	3.9932
Simpsonregel	20	2.000000423	41	-4.23E-7	16.04
	40	2.000000026	81	-2.60E-8	16.27
	80	2.000000002	161	-2.00E-9	-

Wertung

(1) Mit annähernd demselben Aufwand an Funktionsberechnungen K liefert S wesentlich bessere Näherungen als T.

(2) Der Quotient q zweier aufeinanderfolgender Fehler einer Quadraturformel $F_{I_N} = I - I_N$ strebt für $h \to 0$ gegen 2^p mit Fehlerordnung p.

Beispiel 2

$$I = \int_0^2 e^x \, dx = 6.389\,056\,099, \quad h = \tfrac{b-a}{N},\ \bar{h} = \tfrac{h}{n}$$

Unter anderem sehen wir durch die Ergebnisse in der Tabelle folgende Fehlerrelationen bestätigt.

$$\begin{aligned} F_{R_L} : F_{R_M} &= \tfrac{h}{2} : \tfrac{h^2}{24} = 96 \\ |F_T| : F_{R_M} &= \tfrac{h^2}{12} : \tfrac{h^2}{24} = 2 \\ F_T : F_S &= \tfrac{h^2}{12} : \tfrac{(2h)^4}{2880} = \tfrac{\bar{h}^2}{12} : \tfrac{\bar{h}^4}{180} = 960 \end{aligned}$$

Regel	Anzahl N	n	Anzahl der FW-Berechn.	$I - I_N$
R_L	16	1	16	0.39
R_R	16	1	16	-0.40
R_M	16	1	16	0.0041
$T \equiv N_2$	16	1	17	-0.0083
$S \equiv N_3$	8	2	17	-0.000 008 6
N_4	5	3	16	-0.000 025 1
N_5	4	4	17	-0.000 000 051
N_6	3	5	16	-0.000 000 161

2.7.3 Die MATLAB Funktion `trapz`

MATLAB stellt für die numerische Berechnung von bestimmten Integralen die einfache bzw. zusammengestzte Trapezregel mit dem m-File `trapz` zur Verfügung. Die Aufrufe erfolgen mit der Referenz als Vektoren $x(1:n+1)$, $y(1:n+1)$ und sind

```
z = trapz(x,y)
z = trapz(y)
```

Im ersten Fall wird ein Näherungswert im Intervall $[x_1, x_{n+1}]$ berechnet

$$z = \sum_{i=1}^{n} (x_{i+1} - x_i)\ \tfrac{1}{2}(y_i + y_{i+1}).$$

Die zweite Anwendung bedeutet die Integration über das Standardintervall $[0, n]$ mit $n+1$ äquidistanten Stützstellen, das bedeutet n Teilintervalle der Länge 1, und den zugehörigen Stützwerten y_i

$$z = \sum_{i=1}^{n} \tfrac{1}{2}(y_i + y_{i+1}), \quad h = 1.$$

Bei gleichabständigen Stützstellen im Intervall $[a, b]$ gilt wegen

$$\int_a^b f(x)\ dx = \sum_{i=1}^{n} \int_{x_i}^{x_i+h} f(x)\ dx = \sum_{i=1}^{n} h \int_0^1 f(x_i + ht)\ dt \approx h \sum_{i=1}^{n} \frac{1}{2}(y_i + y_{i+1})$$

die Beziehung `trapz(x,y) = h*trapz(y)`.

```
function z = trapz(x,y)
%TRAPZ Trapezoidal numerical integration.
% Z = TRAPZ(X,Y) computes the integral of Y with respect to X using
% trapezoidal integration.  X and Y must be vectors of the same length,
% or X must be a column vector and Y a matrix with as many rows as X.
```

```
% TRAPZ computes the integral of each column of Y separately.
% The resulting Z is a scalar or a row vector.
%
% Z = TRAPZ(Y) computes the trapezoidal integral of Y assuming unit
% spacing between the data points.  To compute the integral for
% spacing different from one, multiply Z by the spacing increment.
%
% See also SUM, CUMSUM.

%  Clay M. Thompson, 10/16/90; Cleve Moler, 1/19/92.
% Copyright (c) 1984-94 by The MathWorks, Inc.

%   Make sure x and y are column vectors, or y is a matrix.
if nargin < 2, y = x; end
[m,n] = size(y);
if m == 1, y = y(:); m = n; end
if nargin < 2, x = 1:m; end
x = x(:);
if length(x) ~= m
    error('Input arguments must be the same length.');
end

%   Trapezoid sum computed with vector-matrix multiply.

z = diff(x)' * (y(1:m-1,:) + y(2:m,:))/2;

% Ende TRAPZ
```

Beispiel

Bestimme näherungsweise das Integral $I = \int_0^1 \frac{dx}{x+1}\, dx = \ln 2$ bei Einteilung des Integrationsintervalls in $n = 2, 4, 8$ Teile.

1. Verwende die Rechteck-, Trapez- und Simpsonregel.
2. Gebe den Integrationsfehler an.
3. Diskutiere die Ergebnisse bei vergleichbarem numerischen Aufwand.

```
disp('Numerische Integration')
fy = '1/(x+1)'
int(fy)
int(fy,0,1)
intf = numeric(int(fy,0,1))
```

```
disp('1) Verschiedene zusammengesetzte Quadraturen')
disp('Trapezregel mit trapz')
ff = '1./(x+1)'     % Vektorargument
n = 2
xx = linspace(0,1,n+1)
yy = numeric(subs(ff,xx,'x'))
z = trapz(xx,yy)     % genauere Quadraturformeln: quad,quad8

disp('Trapezregel mit trapz und Verfeinerung')
for k=1:3
  n = 2^k;
  xx = linspace(0,1,n+1);
  yy = numeric(subs(ff,xx,'x'));
  z = trapz(xx,yy)-intf;
  disp(['n =',sprintf('%1g',n),'   Fehler T-I = ',sprintf('%8e',z) ])
end;
disp('Fehler auf ein Viertel bei n->2*n')

% 3 Verfahren im Vergleich
disp('3 Integrationsregeln mittels Laufanweisungen')
disp('Rechteck, Trapez, Simpson')
nmax = 3;
a = 0;
b = 1;
rr = zeros(nmax,7);
k = 1:nmax;
nv = 2.^k;
rr(:,1) = nv';
for j=1:nmax
  su = 0;
  h = (b-a)/nv(j);
  for i=0:nv(j)-1
    su = su+numeric(subs(fy,a+i*h,'x'));
  end;
  su = su*h;
  rr(j,2) = su;
  rr(j,3) = su-intf;

  su = 0.5*(numeric(subs(fy,a,'x'))+numeric(subs(fy,b,'x')));
  for i=1:nv(j)-1
    su = su+numeric(subs(fy,a+i*h,'x'));
  end;
  su = su*h;
  rr(j,4) = su;
  rr(j,5) = su-intf;
```

```
  su = numeric(subs(fy,a,'x'))+numeric(subs(fy,b,'x'));
  for i=1:2:nv(j)-1
    su = su+4*numeric(subs(fy,a+i*h,'x'));
  end;
  for i=2:2:nv(j)-2
    su = su+2*numeric(subs(fy,a+i*h,'x'));
  end;
  su = su*h/3;
  rr(j,6) = su;
  rr(j,7) = su-intf;
end;

disp('2) Fehlertableau')
disp(['Exaktes Integral = ',num2str(intf)])
disp(' ')
disp('                                    I(n) ')
disp('      n         Rechteck            Trapez              Simpson ')
disp('                Erg     Fehler      Erg     Fehler      Erg     Fehler')
disp(rr)
disp(' ')
disp('3) Anzahl der FW-Berechnungen fuer I(n) : ca n+1')
```

Ergebnisse

```
Numerische Integration
fy =
  1/(x+1)

ans =
  log(x+1)

ans =
  log(2)

intf =
    0.6931

1) Verschiedene zusammengesetzte Quadraturen
Trapezregel mit trapz

ff =
  1./(x+1)

n =
     2
```

```
xx =
         0    0.5000    1.0000

yy =
    1.0000    0.6667    0.5000

z =
    0.7083

Trapezregel mit trapz und Verfeinerung

n =2   Fehler T-I = 1.518615e-002
n =4   Fehler T-I = 3.876629e-003
n =8   Fehler T-I = 9.746698e-004
Fehler auf ein Viertel bei n->2*n

3 Integrationsregeln mittels Laufanweisungen
Rechteck, Trapez, Simpson

2) Fehlertableau
Exaktes Integral = 0.6931
```

	I(n)					
n	Rechteck		Trapez		Simpson	
	Erg	Fehler	Erg	Fehler	Erg	Fehler
2.0000	0.8333	0.1402	0.7083	0.0152	0.6944	0.0013
4.0000	0.7595	0.0664	0.6970	0.0039	0.6933	0.0001
8.0000	0.7254	0.0322	0.6941	0.0010	0.6932	0.0000

```
3) Anzahl der FW-Berechnungen fuer I(n) : ca n+1
```

Die Auswahl der Anzahl n der Teilintervalle bedeutet jeweils Intervallhalbierung.
Die Konvergenzordnung 1 der zusammengesetzten Rechteckregel bzw. 2 der Trapezregel wird durch die Halbierung bzw. Viertelung der Integrationsfehler sichtbar.
Ein Vergleich der Trapezregel und Simpsonregel bei gleicher Anzahl der Knoten $n+1$ führt auf die Betrachtung der Fehlergrößen $c_1\frac{h^2}{12}$ und $c_2\frac{(2h)^4}{2880} = c_2\frac{h^4}{180}$.
Das heißt, wäre $h = 1$, dann ist die Simpsonregel ungefähr 15 Mal genauer.
Für kleines h hat die h-Ordnung der Simpsonregel für den Genauigkeitsvorteil eine größere Bedeutung.

2.8 Asymptotische Fehlerschätzung nach dem RUNGE-Prinzip

Zur Berechnung von I wird eine NC-Formel angewandt.

- Mit Schrittweite h (Feinrechnung), sie liefert die Näherung I_F.
- Mit Schrittweite $2h$ (Grobrechnung), sie liefert die Näherung I_G.

Mit diesen 2 Näherungen I_F und I_G läßt sich eine ungefähre Angabe über den *Fehler der Feinrechnung* $F_F = I - I_F$ machen.

RUNGE-Prinzip

Sei $f(x) \in C^{p+2}[a,b]$ und $I(h)$ der mit der NC-Formel der Fehlerordnung $p \in \mathbb{N}$ errechnete Näherungswert für das Integral I, d.h. $I(h) = I + \mathcal{O}(h^p)$.
$I_F = I(h)$, $I_G = I(2h)$ seien die mit Fein- bzw. Grobrechnung erhaltenen Näherungen für den Integralwert I.

(1) Für den Fehler der Feinrechnung $F_F = I - I_F$ gilt

$$F_F = \frac{I_F - I_G}{2^p - 1} + \mathcal{O}(h^{p+2}).$$

(2) Für den exakten Integralwert I gilt

$$I = I_F + F_F = \frac{2^p I_F - I_G}{2^p - 1} + \mathcal{O}(h^{p+2}).$$

Folgerungen

(1) Beide Beziehungen stellen *asymptotische Schätzungen* dar, d.h. je kleiner die Schrittweite h, desto genauer wird der Wert F_F bzw. I approximiert - falls die Rundungsfehler vernachlässigt werden können.

(2) Für den Fehler der Feinrechnung F_F liefert

$$est = \frac{I_F - I_G}{2^p - 1}$$

i.a. eine gute Näherung.

(3) Extrapolation
Der aus Fein- und Grobrechnung zusammengesetzte (extrapolierte) Wert

$$I_{FG} = \frac{2^p I_F - I_G}{2^p - 1} = I_F + \frac{I_F - I_G}{2^p - 1} = I_F + est$$

ist eine (um Größenordnung) bessere Näherung als der Feinwert I_F.

Übersicht zu den Vorfaktoren bei der Integrationsfehlerermittlung

Integrationsregel	Genauigkeit $\mathcal{O}(h^p)$	Vorfaktor $\frac{1}{2^p-1}$
R_L, R_R	$\mathcal{O}(h)$	1
$T \equiv N_2, R_M = O_1, O_2$	$\mathcal{O}(h^2)$	$\frac{1}{3}$
$S \equiv N_3, N_4, O_3, O_4$	$\mathcal{O}(h^4)$	$\frac{1}{15}$
N_5, N_6	$\mathcal{O}(h^6)$	$\frac{1}{63}$
N_7, N_8	$\mathcal{O}(h^8)$	$\frac{1}{255}$

Beispiel

Zu bestimmen ist näherungsweise der Integralwert

$$I = \int_0^1 \frac{dx}{x+1} = \ln 2 = 0.693147180559 \quad \text{mit} \quad N = 2, 4, 8 \quad \text{Teilintervallen.}$$

Dabei kontrollieren wir, inwieweit der Fehler der zusammengesetzten Quadraturformel (Feinrechnung) mit der Schätzung $est = \dfrac{I_F - I_G}{2^p - 1}$ gemäß dem Runge-Prinzip korrespondiert.

Regel	$N=1$ $h=1$	$N=2$ $h=1/2$	$N=4$ $h=1/4$	$N=8$ $h=1/8$	F_F
R_L	1	0.833333	0.759524	0.725372	
$F_{R_L} = I - R_L$	-0.306853	-0.140186	-0.066377	-0.032225	$\sim h$
F_{R_L} (Runge $p=1$)		-0.166667	-0.073809	-0.034152	
Funktionswerte	1	2	4	8	
R_R	0.5	0.583333	0.634524	0.662872	
$F_{R_R} = I - R_R$	0.193147	0.109814	0.058623	0.030275	$\sim h$
F_{R_R} (Runge $p=1$)		0.083333	0.051191	0.028348	
Funktionswerte	1	2	4	8	
R_M	0.666667	0.685714	0.691220	0.692661	
$F_{R_M} = I - R_M$	0.026480	0.007433	0.001927	0.000486	$\sim h^2$
F_{R_M} (Runge $p=2$)		0.006349	0.001835	0.000480	
Funktionswerte	1	2	4	8	
T	0.75	0.708333	0.697024	0.694122	
$F_T = I - T$	-0.056853	-0.015186	-0.003877	-0.000975	$\sim h^2$
F_T (Runge $p=2$)		-0.013889	-0.003769	-0.000967	
Funktionswerte	2	3	5	9	
S	0.694444	0.693254	0.6931545	0.6931477	
$F_S = I - S$	-0.001297	-0.000107	-0.0000073	-0.0000005	$\sim h^4$
F_S (Runge $p=4$)		-0.000079	-0.0000066	-0.0000005	
Funktionswerte	3	5	9	17	

Bemerkungen

(1) Fehlerordnung

Durch den Vergleich der Fehler erkennt man deutlich die theoretisch festgestellte Fehlerordnung jeder Formel.
In der Tabelle sind die Formeln nach wachsender Genauigkeitsordnung aufgeführt.

(2) Genauigkeitsordnung

Um die Genauigkeit der Formeln zu vergleichen, muß auf etwa diesselbe Anzahl von Funktionswertberechnungen (FW) Bezug genommen werden. Ausgehend vom gleichen Aufwand kann dann eingeschätzt werden, welche Formel günstiger ist. So haben wir z.B. bei 9 FW von T und S einen sehr deutlichen Vorteil der Simpsonregel.

(3) Obwohl die Mittelpunktregel ein etwas kleineres Fehlerglied als die Trapezregel hat, wird sie wegen der Verwendung der Funktionswerte $f(x_i + \frac{h}{2})$ nicht bevorzugt.

2.8.1 Linearkombination von Quadraturformeln

Die Fehlerglieder der Quadraturformeln zeigen noch einen anderen Sachverhalt. Man kann sie einfach vergleichen.
Für gleiches N (vergleichbares h) gilt mit großer Genauigkeit

$$2F_{R_M} \;=\; -F_T.$$

Dies nutzen wir für die Konstruktion einer verbesserten Quadraturformel.
Gehen wir aus von den zusammengesetzten Regeln.

$$\begin{aligned} F_{R_M} &= I - R_M = (b-a)\frac{h^2}{24}f''(0) + \mathcal{O}(h^4) \\ F_T &= I - T = -(b-a)\frac{h^2}{12}f''(0) + \mathcal{O}(h^4) \\ 2F_{R_M} + F_T &= 3I - (2R_M + T) = \mathcal{O}(h^4) \\ \frac{1}{3}(2F_{R_M} + F_T) &= I - \frac{1}{3}(2R_M + T) = \mathcal{O}(h^4) \end{aligned}$$

Der Fehler $2F_{R_M} + F_T$ ist um 2 Potenzen der Schrittweite kleiner als die Größen selbst. Weiterhin erkennen wir, daß wir durch Linearkombination von 2 Integrationsformeln eine neue Formel höherer Genauigkeit gewinnen.

$$I = \frac{1}{3}(2R_M + T) \;+\; \mathcal{O}(h^4)$$

2.8.2 Die MATLAB Funktionen `quad, quad8`

Die Simpsonregel bez. die NC-Formel N_8 finden Eingang in die numerischen rekursiven adaptiven Algorithmen zur Quadratur `quad, quad8`.
Dabei wird das Runge-Prinzip der Grob- und Feinrechnung mit einer Steuerung der lokalen Integrationsschrittweite bei einzuhaltender Toleranz angewandt.

3 Anhang

Anhang A

Zusammenstellung von Adressen

1. World Wide Web (WWW) mit MATLAB Seiten

Die TEX Quelle sowie das PostScript file `primer35.ps` der 2.Edition des MATLAB Primers steht stehen mittels *ftp* auf `ftp.math.ufl.edu` im Verzeichnis `pub/matlab` zur Verfügung.

Die MathWorks Inc. und MathTools Ltd. entwickeln und vertreiben die Computersoftware MATLAB und andere Komponenten und sind natürlich mit ihrem ganzen Angebot auch im Internet zu finden.

http://www.mathworks.com/	The MathWorks Inc. home (auch Simulink)
http://www.mathworks.com/products/matlab/	MATLAB 5.3
http://www.mathworks.com/products/matlab/	MATLAB web server 1.0
http://www.mathworks.com/support/books/	MATLAB based books
http://www.mathtools.com/	MATLAB Toolboxen von MathTools Ltd. (auch MATCOM, MIDEVA)
http://krum.rz.uni-mannheim.de/cafgbench.html	Computer Algebra Benchmarks (RZ/Uni Mannheim)

2. Verzeichnisse mit MATLAB m-Files für Skripte und Funktionen

Zu den Kapiteln 1-2 des Skripts liegen die entsprechenden Files (meist m-Files) und diverse Daten- und Ergebnisfiles vor.

`*.m, *.ps, *.eps, *.mat`

Die Dateien sind zu finden im Novell-Netz PIVOT des Instituts für Mathematik bzw. auf der persönlichen Homepage im Internet.

\\PIVOT\SHARE Q:\NEUNDORF\STUD_M93\MATLAB4

Homepage Navigator → Publications → Computeralgebra → MATLAB4

e-mail: ***neundorf@mathematik.tu-ilmenau.de***

Homepage: ***http://imath.mathematik.tu-ilmenau.de/~neundorf/index_de.html***

Literatur

[1] Sigmon, K.: *MATLAB Primer, Second Edition.*
Department of Mathematics, University of Florida USA 1992.

[2] Sigmon, K.: *MATLAB Primer 5e.* CRC Press 1998.

[3] *MATLAB User's Guide and Reference Guide.*

[4] *MATLAB Release Notes 4.1. For Unix Workstations.*
The MathWorks Inc. Natick, Massachusetts USA 1993.

[5] *The Student Edition of MATLAB. Version 4. User's Guide.*
The MathWorks Inc. Prentice Hall, Englewood Cliffs New Jersey 1995.

[6] *The Student Edition of MATLAB 5.* Prentice Hall.

[7] Redfern, D.; Campbell, C.: *The MATLAB 5 Handbook.* Springer-Verlag 1998.

[8] Köckler, N.: *Numerische Algorithmen in Softwaressystemen: unter besonderer Berücksichtigung der NAG-Bibliothek.* B.G. Teubner Stuttgart 1990.

[9] Neundorf, W.: *MATLAB - Teil I: - Vektoren, Matrizen, lineare Gleichungssysteme.* Preprint M 20/99 IfMath der TU Ilmenau, Juli 1999.

[10] Neundorf, W.: *MATLAB - Teil II: - Speicheraspekte, spezielle LGS, SDV, EWP, Graphik, NLG, NLGS.* Preprint M 23/99 IfMath der TU Ilmenau, September 1999.

[11] Neundorf, W.: *MATLAB - Teil III: - Komplexe LGS, Interpolation, Splines.* Preprint M 10/00 IfMath der TU Ilmenau, Mai 2000.

[12] Mathews, J.H.; Fink, K.D.: *Numerical Methods using MATLAB.* Prentice Hall London 1999.

[13] Chen, K.; Giblin, P.J.; Irving, A.: *Mathematical explorations with MATLAB.* Cambridge University Press 1999.

[14] Mohr, R.: *Numerische Methoden in der Technik: eine Lehrbuch mit MATLAB-Routinen.* Vieweg Braunschweig 1998.

[15] Golubitsky, M.; Dellnitz, M.: *Linear algebra and differential equations using MATLAB.* Brooks/Cole Pub. Co Pacific Grove 1999.

[16] Maeß, G.: *Vorlesungen über numerische Mathematik II.* Akademie-Verlag Berlin 1988.

Anschrift:

Dr. Werner Neundorf
Technische Universität Ilmenau, Institut für Mathematik
PF 10 0565
D - 98684 Ilmenau

e-mail : neundorf@mathematik.tu-ilmenau.de